BENEFIT MAXIMIZING CLASSIFICATION USING FEATURE INTERVALS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Nazlı İkizler

September, 2002

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. H. Altay Güvenir (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Cevdet Aykanat

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Özgür Ulusoy

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray

Director of the Institute

ABSTRACT

BENEFIT MAXIMIZING CLASSIFICATION USING FEATURE INTERVALS

Nazlı İkizler
M.S. in Computer Engineering
Supervisor: Prof. Dr. H. Altay Güvenir
September, 2002

For a long time, classification algorithms have focused on minimizing the quantity of prediction errors by assuming that each possible error has identical consequences. However, in many real-world situations, this assumption is not convenient. For instance, in a medical diagnosis domain, misdiagnosing a sick patient as healthy is much more serious than its opposite. For this reason, there is a great need for new classification methods that can handle asymmetric cost and benefit constraints of classifications. In this thesis, we discuss cost-sensitive classification concepts and propose a new classification algorithm called *Benefit Maximization with Feature Intervals* (BMFI) that uses the feature projection based knowledge representation. In the framework of BMFI, we introduce five different voting methods that are shown to be effective over different domains. A number of generalization and pruning methodologies based on benefits of classification are implemented and experimented. Empirical evaluation of the methods has shown that BMFI exhibits promising performance results compared to recent wrapper cost-sensitive algorithms, despite the fact that classifier performance is highly dependent on the benefit constraints and class distributions in the domain. In order to evaluate cost-sensitive classification techniques, we describe a new metric, namely *benefit accuracy* which computes the relative accuracy of the total benefit obtained with respect to the maximum possible benefit achievable in the domain.

Keywords: machine learning, classification, cost-sensitivity, benefit maximization, feature intervals, voting, pruning.

ÖZET

ÖZNİTELİK ARALIKLARIYLA FAYDA MAKSİMİZASYONUNA YÖNELİK SINIFLANDIRMA

Nazlı İkizler
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. H. Altay Güvenir
Eylül, 2002

Uzun zamandır, sınıflandırma algoritmaları bütün olası hataların sonuçlarının benzer olacağı varsayımıyla, tahmine dayalı hataların sayısını azaltma üzerinde yoğunlaşmışlardır. Fakat bu varsayım, gerçek hayattaki pek çok durum için elverişli değildir. Örneğin, tıbbi tanı alanında, hasta olan bir kimseyi sağlıklı olarak sınıflandırmak tam tersi duruma oranla çok daha ciddi bir hatadır. Bu nedenle, sınıflandırmaların bu tip asimetrik maliyet ve fayda kriterlerini göz önünde bulunduracak yeni sınıflandırma metotlarına büyük ihtiyaç vardır. Bu tezde, maliyete duyarlı sınıflandırma kavramları üzerinde durulmakta ve öznitelik izdüşümü tabanlı bilgi gösterimini kullanan, *Öznitelik Aralıklarıyla Fayda Arttırma* (BMFI) olarak isimlendirilen yeni bir sınıflandırma algoritması sunulmaktadır. BMFI çatısı altında, farklı veri kümelerinde etkili olduğu gösterilen beş ayrı oylama yöntemi tanıtılmıştır. Bununla birlikte, birkaç genelleme ve budama yöntemi geliştirilmiş ve denenmiştir. Deneysel değerlendirmelerde BMFI, performansın problemin veri kümesindeki fayda kriterlerine ve sınıf dağılımlarına çok bağlı olması gerçeğine rağmen, sarma prensibine dayalı yeni maliyete duyarlı sınıflandırma algoritmalarıyla karşılaştırıldığında, umut verici bir performans sergilemiştir. Ek olarak, maliyete duyarlı ve fayda arttırımına yönelik metotların değerlendirilmesi için, *fayda doğruluğu* olarak isimlendirilmiş yeni bir metrik önerilmiştir. Bu metrik, elde edilen toplam faydanın, mümkün olan en yüksek fayda değerine oranla göreceli doğruluğunu hesaplamaktadır.

Anahtar sözcükler: makine öğrenmesi, sınıflandırma, maliyet duyarlılığı, fayda maksimizasyonu, öznitelik aralıkları, oylama, budama.

Acknowledgements

Words can never be enough in expressing how grateful I am to many incredible people in my life who made this thesis possible. First of all, I am deeply indebted to my supervisor Prof. Dr. H. Altay Güvenir, who has guided me with his invaluable suggestions, lightened up the way in my darkest times and encouraged me a lot in the academic life. It was a great pleasure for me to have a chance of working with him.

I would like to address special thanks to Prof. Dr. Cevdet Aykanat and Assoc. Prof. Özgür Ulusoy, for accepting to read and review this thesis. I would also like to acknowledge the financial support of TÜBİTAK (Scientific and Technical Research Council of Turkey) under the grant 101E044 for this research.

Probably most of this work would not have been possible without the technical and emotional support of dear Engin Demir. I owe him a lot.

I would like to thank all the people of the room EA526, especially to Ediz Şaykol, for their caring friendship, motivation and profound assistance. I am grateful to all of my friends who fulfilled my life with joy.

Above all, I owe everything to my parents, who supported me in each and every way, believed in me permanently and inspired me in all dimensions of life. Without their everlasting love, this thesis would never be completed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the most important characteristics of human brain is its capability to learn from experience, manipulate the gained knowledge and to make use of it in forecasting possible future events. This learning process is crucial for human being since it is the doorway to innovation and advancement. For this reason, from the evolution of computers, researchers try to mimic the way brain works and to integrate various qualifications of intelligence to the computer. Computer scientists in the field of machine learning are the foremost people dealing with such issues.

Machine learning is the research area that seeks to build systems that interpret the data compiled in the datasets or the perceptions collected from the environment for understanding and making use of the knowledge beneath. Machine learning techniques are being investigated on and applied to various problems such as natural language processing, handwriting and speech recognition, text and document categorization, knowledge discovery in databases, remotely-sensed image analysis, medical data analysis and diagnosis, weather prediction, email filtering and various applications on the World Wide Web domain.

In the last few years, significant practical achievements have been obtained in learning systems by taking advantage of several established learning algorithms.

Bayesian methods, decision trees, neural networks, instance-based learning algorithms, support vector machines and genetic algorithms are among those powerful methods which have aided the learning community in practical applications.

The *learning problem* is the task of finding general rules that explain data given only a sample of limited size [25]. In this concept, consider a child who is learning to speak. He is imposed to a flow of sounds and images from his environment, and apparently, this environment is a limited portion of the real world. What he does is, by using perceptions, to acquire attributes of the items around him and to form an association between the items and the expressions his parents use simultaneously. In this framework, color, shape and smell of an item are the foremost characteristics that the child observes. His parents' word concerning the item is the name of the item. By combining these inputs, he learns the name of the object. Since he has been leaded to that result by his parents' assistance, he is said to learn under the supervision of his parents. Counterpart of this situation in machine learning terminology is *supervised learning*.

In this study, we explore the directions of decision making under supervised learning framework and try to find ways to optimize consequences of predictions in the real-world situations.

1.1 Motivation

Life is a combination of decisions. Outcomes of these decisions can either be good or perfect, bad or terrible depending on the correctness of choices. For instance when you decide on where to invest your money, there is a bunch of possibilities. You may invest your money in a bank and earn a comparably low yet regular amount of interest. Or, you can buy stocks of exchange and gain more money, however, in such a circumstance, there is a major risk of losing all the money you put in. Your net profit depends on how clever and logical your decision of investment is. In another incident, suppose that you are the head of an emergency desk in the hospital and two patients come along. You have limited source of instruments and you have to decide which patient to examine first. Your judgment in such a case is vital from the patients' point of view. Hence, there is a scale

for outcomes of decisions made, there can be minor or life-threatening mistakes, and there can be slight achievements or major successes that can change a person's life.

Keeping this fact in mind, when we look at typical machine learning applications of present-day, algorithms hardly evaluate the effectiveness and applicability of their decisions. In classical machine learning applications, what the algorithms try to accomplish is to reduce the quantity of the error obtained and, most of the time, the quality of the error is ignored. However, as the above examples demonstrate, the characteristics of errors can be vital. For this reason, before taking an action, consequences of decisions should be elaborated and investigated extensively.

The brand new subfield of machine learning that evaluate the effects of decisions is *cost-sensitive machine learning*. It is based on incorporating the so-called cost knowledge to the process of classification. Costs can be categorized under many headings such as costs of collecting data, acquiring features or costs of misclassifications. The most crucial of these costs are misclassification costs and in this study, we will be concentrating on evaluation with respect to misclassification costs.

In this thesis, we propose a cost-sensitive machine learning technique that represents the learned information in the form of feature projections of the training instances. Classification algorithms that use this knowledge representation scheme are called *Feature Projection Based Classifiers* and many such classifiers have been shown to be quite successful in a wide range of real-world problem domains ([14], [28], [29] and [30]). In this study, we introduce another feature projections based classification algorithm, called Benefit Maximizing classifier with Feature Intervals (BMFI, for short). Voting procedure of CFI in [28] has been changed to impose the cost-sensitive property to the algorithm. A number of generalization and pruning techniques have been utilized. BMFI, along with its versions containing pruning have been evaluated over several benchmark datasets from UCI Machine Learning repository and several real world datasets, especially on a financial dataset with loan applications.

1.2 Overview of the Thesis

Chapter 2 of the thesis presents fundamental concepts about supervised learning. Cost-sensitive learning is introduced in this chapter together with the types of costs considered. Cost matrix and benefit matrix notions are evaluated in detail and several algorithmic definitions that will be used throughout the thesis are presented. Feature dependent benefit problem is another headline from Chapter 2.

In Chapter 3, several different approaches to misclassification cost-sensitive learning in the literature are described and discussed thoroughly. Two main groups of algorithms are presented within this context: wrapper algorithms and direct cost-minimizing algorithms.

Chapter 4 hosts the algorithmic descriptions of our BMFI algorithm along with the details of feature projection concept, voting methods and cost-sensitivity elements. Generalization and pruning methodologies are presented. Illustrative examples delineate the progression of the algorithm clearly.

Experimental evaluation of the proposed algorithm is presented in Chapter 5 by the results of its application to real world and benchmark datasets. Its comparison to MetaCost and cost-sensitive classifier on Weka over Naive Bayesian Classifier, C4.5 decision tree learner and VFI classifier is also included in this chapter.

Chapter 6 reviews the results and the contributions of this thesis and outlines future research directions on this subject.

Chapter 2

Cost and Benefit

In this chapter, firstly supervised learning or inductive concept learning, with its basic terminology is introduced. An outline of the different types of cost considered in inductive learning is presented. Subsequently, we sketch the borderlines of cost-sensitive learning along with its fundamental concepts. Finally, the need for a benefit matrix representation is discussed and non-stationary benefit domains are explored.

2.1 Supervised Learning

In the context of supervised learning, the instances provided to the learning program have class labels associated with them. For this reason, supervised learning is also called induction from examples. The aim is to produce a classifier capable of predicting labels of the unseen cases correctly.

More formally, given a set of labeled examples $<x_i,y_i>$ where x_i is a vector of continuous or discrete values called *features* and y_i is the *label* of x_i, *supervised learning* is finding a mathematical model that accurately labels a high proportion of unlabeled examples drawn from the same probabilistic distribution.

The input set of examples is called *training data* or *instance space* and it is assumed to inherit an unknown probability distribution $P(x,y)$ of the class labels. Features can have either *linear* (*discrete or continuous*) or *nominal* (categorical) values. For example, "age of the patient" in a medical domain dataset is a linear feature which can have discrete values from a subset of integers. On the other hand, "exchange rate of US Dollar to Turkish Lira" is a continuous linear feature which is assessed from a subset of real numbers. Conversely, "color" is a nominal feature possessing values from a predefined range of color attributes.

Similar to the feature categorization, labels y_i, i.e., classes of the instances in the dataset can either be elements of a discrete set of classes such as $\{1,2,...,N\}$ or elements from a continuous set such as real numbers. When the set of possible predictions is discrete, the supervised learning procedure is called *classification* or *concept learning*. On the other hand, if possible predictions can be drawn from a continuous subset of values; this task is called *regression* or *function approximation.* Besides, instances in a dataset can be assigned more than one class label depending on the nature of the problem. Such a learning problem is called *multi-class classification*. In this thesis, we will be focusing on the single-class classification problem in which there is only one class assigned to each instance.

In order to determine the predictive capability of a learning system, an independent *test data* that was not used at any time during the learning process is presented to the model. This test data is a set of unlabeled examples, i.e., $<x_i>$'s, assumed to possess the same probability distribution $P(x,y)$ as the training set. In most of the classification systems, the metric used for evaluating a model's predictive capacity is the *accuracy* of the system. Accuracy is defined as the rate of correct predictions made by the model over the data set [34]. It refers to the degree of fit between the model and the data.

2.2 Cost-Sensitive Learning

In traditional classification systems, all types of errors are treated in the same manner and predictive accuracy of the system simply measures the ratio of correct predictions.

However, in many real-world domains, errors may differ in significance and may have different consequences. An obvious example of this situation is available in the medical diagnosis domain. Misdiagnosing a patient who is ill as being healthy has much more serious consequences than misdiagnosing a patient who is healthy as having a disease. Hence, in such situations, it is not enough to simply predict the most probable class. Instead, the system should predict in a way to minimize unwanted side effects, namely costs.

Therefore, traditional classification systems mostly seem to fail in real world domains where correct and incorrect classifications have different interpretations. That is why cost-sensitive classification systems are being recently studied. The goal of these classification schemes is to minimize the total cost acquired by the prediction process. Since conventional predictive accuracy metric does not include cost information, it is possible for a less accurate classification model to be more cost-effective in reality. This means, to obtain the minimal cost, cost-sensitive learning systems may need to trade off some of the predictive accuracy and are subject to make more mistakes in quantity.

2.2.1 Types of Cost in Supervised Learning

Turney has created a taxonomy of the different types of cost in inductive concept learning in [46]. According to this taxonomy there are nine major types of costs. Some of these types can be overviewed as follows:

- *Cost of misclassification errors*: This type of errors is the most crucial one and most of the cost-sensitive learning research has investigated the ways to manipulate such costs. These error costs can either be constant or conditional depending on the nature of the domain. Conditional misclassification costs may depend on the characteristics of a particular case, on time of classification, on feature values or on classification of other cases.

- *Cost of tests (features)*: In some domains, such as medical diagnosis, some of the tests (i.e., features) may have acquirement costs. For instance, taking a

computational tomography is a costly operation and doctors avoid prescribing unless it is especially required. This necessity is proportional to the cost of misclassification. If the cost of misclassification surpasses the costs of tests greatly, then all tests of predictive value should be taken into consideration. Similar to error costs, test costs can be constant or conditional on various issues such as prior test selection and results, true class of instance, side effects of the test or time of the test.

- *Cost of teacher*: It might be expensive to determine the correct class of an example in some circumstances. In such a case, a learning algorithm should rationally try to minimize the cost of teaching, and one possible way is actively selecting instances for the teacher, i.e., active learning. Again, this type of costs can be constant or varying dependent on individual cases.

- *Cost of computation*: Size and structural complexity, time and space requirements of a classification algorithm both in training and test phases are considered under this category.

- *Cost of cases*: Turney states that there may also be a cost of acquiring instances [46]. In such situations, it is argued that cost of cases for a batch learner and an incremental learner should be evaluated separately.

In addition to these types, there may be other kind of costs such as intervention costs, unwanted achievement costs, human-computer interaction costs and costs of instability. Nevertheless, most of these costs are non-trivial and hard to formulate since they are generally domain dependent and irregular. In our studies, we concentrated on costs of misclassification and in this thesis, we use the term cost related to this type of error costs.

2.3 The Cost Matrix

Definition 2.1: *C=[cij] is a n×m cost matrix of domain D if n equals to the number of prediction labels, m equals to the number of possible class labels in D and c_{ij}'s are such that*

$$c_{ij} = \begin{cases} 0 & \text{if } i = j \\ >0 & \text{if } i \neq j \end{cases}$$

According to Definition 2.1, a square cost matrix of order *n* has the following structure:

		Actual			
Prediction	C0	C1	.	.	Cn
C0	c_{00}	c_{01}	.	.	c_{0n}
C1	c_{10}	c_{11}			c_{1n}
.	.				.
.	.				.
Cn	c_{n0}	c_{n1}	.	.	c_{nn}

where rows of the matrix correspond to predicted classes and the columns of the matrix correspond to actual classes. Thus, c_{ij} represents the cost of classifying an instance of class *j* as class *i*.

In the cost matrix formation, the elements $c_{00}, c_{11}, c_{22},\ldots,c_{nn}$ which constitute up the main diagonal of the matrix are assumed to be all 0, representing the natural interpretation that correct classifications have no cost to the user. On the other hand, the non-diagonal elements of the cost matrix are assumed to be greater than zero, denoting loses of misclassification from a positive baseline. However, this positive representation of costs is far form the natural perception of net gain flow concept, as we will see shortly.

When there are *n* probable classes in classification and the algorithm forces a class to be determined, the cost matrix of classification is a square matrix of order *n*. If there is a probability to leave the instance's class label undetermined by the classification

algorithm, then the cost matrix is a rectangular $(n+1)\times n$ matrix where the extra row stands for possible losses and gains for the undetermined cases. In our evaluations, we omit the undetermined class option and force the classification algorithm to predict a class for each test instance. Hence, in our computations, and from now on in this thesis, we will be talking over $n\times n$ square matrices.

Table 2.1: An example cost matrix of NYNEX MAX domain [40].

	Actual Class		
Prediction	PDF	PDO	PDI
PDF	0	150	250
PDO	100	0	250
PDI	150	50	0

Table 2.1 shows an example cost matrix taken from [40] which denotes the cost matrix for problem of dispatching technicians to fix faults in the local loop of a telephone network (NYNEX MAX domain). This cost matrix represents the costs associated with each of the three dispatches, PDF, PDO, PDI. As it can be seen, the cost matrix is asymmetric and different types of misclassifications have different costs. For example, identifying a PDI dispatch as PDO is five times more costly than dispatching a PDI instead of a PDO. From such a cost matrix, we can see that identification of PDI dispatch is more important from the company's point of view, since its erroneous classification inquires the most cost.

Errors made in a classification algorithm can be viewed as a special case of cost. Specifically, if the cost matrix has uniform cost distribution over all classes, and non-diagonal elements of cost matrix are all 1's, then resultant total cost simply gives the error made by the classification algorithm. Such a cost matrix is called uniform cost matrix [37].

2.3.1 Optimal Prediction Using Cost Matrices

In a cost-sensitive classification problem, an instance should be predicted to have the class label that leads to the lowest expected cost [20]. More formally, the optimal prediction for an example x is the class that minimizes

$$EC(x,i) = \sum_j P(j \mid x)C(i,j) \tag{2.1}$$

where P($j \mid x$) is the probability that x has the true class j, $C(i,j)$ is the cost of predicting class i when the true class of the instance is j and $EC(x,i)$ is the expected cost of prediction (also referred as conditional risk [18]). If $i=j$ then the prediction is correct, if $i \neq j$ then the prediction is incorrect. According to this formulation, although some class k is more probable for an instance x, it can be more optimal to predict another class for the sake of cost minimization.

2.3.2 Reasonableness of the Cost Matrix

Cost matrix logic comes from the natural fact that the cost of correct classification of an instance can never be higher than the cost of incorrect classification. Elkan has named this condition as 'reasonableness condition' and for a two-class cost matrix, he has mathematically formulated it as $c_{10}>c_{00}$ and $c_{01}>c_{11}$ [20].

To generalize this condition to multiple possible classes, we define the reasonableness condition as follows:

Definition 2.2*: An $n \times n$ cost-matrix is reasonable if and only if for each $i,j \in \{0,\ldots,n\}$ and $i \neq j$, $c_{ij}>c_{jj}$.*

When evaluating the predictive capability of a cost-sensitive system, the reasonableness of the cost matrix is a crucial requirement. As pointed out in [36], a cost matrix should let each possible class label be predictable by the cost-sensitive classifier. If a cost matrix is not reasonable, some class labels may never be predicted by the optimal cost-sensitive decision policy. For instance, when for all $C(m,j) \geq C(k,j)$; i.e., all the cost values of row m dominate cost values of row k, optimal decision policy never

predicts class *m*, since there exists a better decision for all possibilities, which is class *k*, that will lead to lesser cost.

Consider the following two examples: In a three-class cost-sensitive classification problem with the cost matrix in Table 2.2, there is no need to run any learning algorithm, since it is obvious that, no matter what the class probability distributions are, the optimal prediction is always C_1.

Table 2.2: Cost matrix for which the optimal prediction is always C_1 and thus no learning is needed.

	Actual class		
Prediction	C_1	C_2	C_3
C_1	1	2	4
C_2	2	3	5
C_3	3	10	7

Similarly, if the cost matrix in use is the one shown in Table 2.3, optimal classifier never makes it choice from C_1, because C_2 and C_3 predictions always outperform in terms of cost [36].

Table 2.3: Cost matrix for which C_1 is never predicted.

	Actual Class		
Prediction	C_1	C_2	C_3
C_1	3	10	7
C_2	2	0	5
C_3	1	3	1

In this thesis, we make sure that all the cost matrices used for evaluation purposes are reasonable and they allow all class labels to be predicted.

2.3.3 Operations on Cost Matrices

Some particular operations can change the baseline of cost matrices from which costs are measured, without changing the optimal predictions made. These operations are useful especially when the unit amount for costs are subject to any change. In this subsection, we present two such elementary operations: scaling and shifting.

2.3.3.1 Scaling

Given a cost matrix C, suppose each entry of the cost matrix is multiplied by a positive constant *b*. In Equation 2.1, each C(*i*,*j*) is multiplied by *b*, so we have

$$E(x,i) = \sum_j P(j \mid x) \times b \times C(i,j) = b \sum_j P(j \mid x) \times C(i,j) \tag{2.2}$$

As the above equivalence shows, we can formulate the optimal decision criterion in terms of the original matrix and since *b* is a constant, the optimal decisions made by the cost-sensitive classifier do not change [20]. The only change is in the total cost obtained in the result of the decisions. This operation is called *scaling* and it can also be interpreted as changing the unit measure of costs.

2.3.3.2 Shifting

In a similar fashion to scaling, when a positive constant is added to each entry of a cost matrix, the optimal decisions made by a cost-sensitive algorithm is unchanged. Shifting operation is useful when we want to represent all the entries of a cost matrix from a different baseline, such as the zero baseline with all costs being positive. As it has been pointed out in [20], this shifting means changing the baseline of cost measurements by the addition of positive constant.

Shifting operation can be formulated as follows: Suppose positive constant b is added to each entry of the cost matrix $C(i,j)$. Then, optimal decision equation (Equation 2.1) is modified as in Equation 2.3:

$$E(x,i) = \sum_j P(j \mid x) \times (C(i,j) + b) = \sum_j [(P(j \mid x) \times C(i,j)) + (P(j \mid x) \times b)]$$
$$= \sum_j P(j \mid x) \times C(i,j) + b \sum_j P(j \mid x) = \sum_j P(j \mid x) \times C(i,j) + b \qquad (2.3)$$

By the nature of probability distributions, $\sum_j P(j \mid x) = 1$. Hence, by Equation 2.3, each expected total cost value is incremented by the positive constant b and optimal decision which chooses the minimum of these values is unchanged.

2.4 Benefit Matrix

Recent research in machine learning has used the terminology of costs when dealing with misclassifications. However, those studies mostly lack the information that correct classifications may have different interpretations. Besides implying no cost, accurate labeling of instances may entail indisputable gains. Elkan points out the importance of these gains [20]. He states that doing accounting in terms of benefits is commonly preferable because there is a natural baseline from which all benefits can be measured, and thus, it is much easier to avoid mistakes.

Benefit concept is more appropriate to real world situations, since net flow of gain is more accurately denoted by benefits attained. If a decision made is profitable from the decision agent's point of view, its benefit is positive. Otherwise, it is negative, which equals to the cost of wrong decision. To incorporate this natural knowledge of benefits to the notion of cost-sensitive learning, in this thesis we have used *benefit matrices* (sometimes referred as *cost-benefit matrices* in literature [1]).

Definition 2.3*: $B=[b_{ij}]$ is a $n \times m$ benefit matrix of domain D if n equals to the number of prediction labels, m equals to the number of possible class labels in D and b_{ij}'s are such that*

$$b_{ij} = \begin{cases} \geq 0 & \text{if } i = j \\ < b_{ii} & \text{if } i \neq j \end{cases}$$

In benefit matrix representation, b_{ij} represents the benefit of classifying an instance of true class j as belonging to class i. Benefit matrix structure is just like the cost matrix, with the extension that entries can either have positive or negative values. In addition, diagonal elements (b_{ii}'s) should be non-negative values, ensuring that correct classifications can never have negative benefits associated with them.

Table 2.4 presents a benefit matrix for a binominal classification problem. In this benefit matrix, misjudgment of an actual "bad" instance as "good" is assigned a negative benefit of 200 whereas correct identification of a "bad" instance has a gain of 100. In this domain, although correct classification of "good" instances has a certain benefit, identification of "bad" instances are 10 times more beneficial.

Table 2.4: An example benefit matrix for a two-class problem.

	Actual class	
Prediction	good	bad
good	10	-200
bad	-50	100

Benefit matrices can also be interpreted as the negation of cost-matrices in which the diagonal elements are non-negative. Thus, a benefit matrix incorporates all the characteristics of a cost matrix, and all operations applicable to cost matrices generate the similar results in benefit matrices. Specifically, benefit matrices should obey reasonableness rule, which is already satisfied by the definition of benefit matrices, and can be subject to scaling and shifting operations without any alteration in the optimal predictions.

In some situations, incorrect classifications can also bring benefits. For example in a medical diagnosis domain, classifying a type of a disease as another type can still be beneficial, rather than classifying the patient as healthy. Of course, this kind of erroneous classifications is never more beneficial than the accurate one, but by further investigations and common treatment techniques, it can be manageable. An example to such a domain is lesion (gastric carcinoma) dataset, which has the benefit matrix approved by experts given in Table B.5 in Appendix B.

Our benefit model resembles the cost model proposed in [15] to some extent. In Domingos's study, the aim is to answer the question of whether a machine learning system should be deployed depending on its net present value (NPV) [9]. To accomplish this goal, the so-called cost model is also formulated in terms of cash flows, instead of costs, asserting the awkwardness of treating revenues as negative costs.

2.4.1 Optimal Prediction Using Benefit Matrices

Using the framework of benefit matrices, the cost-sensitive classification problem is slightly modified to involve benefits. Since costs are negated to represent benefits, minimization problem becomes a maximization problem. Thus, given a benefit matrix B the optimal prediction for an example x is the class that maximizes

$$EB(x,i) = \sum_j P(j \mid x) B(i,j) \tag{2.4}$$

where $P(j \mid x)$ is the probability that x has the true class j, $B(i,j)$ is the benefit of predicting class i when the instance x has true class j and $EB(x,i)$ is the expected benefit of making that prediction.

Equation 2.4 represents the expected benefit in classifying a single instance x. The total expected benefit of the classifier model m over the whole test data is

$$EB_m = \sum_x \arg\max_{i \in Y} EB(x,i) = \sum_x \sum_j P(j \mid x) B(i,j) \tag{2.5}$$

2.4.2 Cost and Benefit Matrix Equivalence

In [37] it has been shown that a benefit matrix can be transformed into a cost matrix by using the following theorem.

Definition 2.4: *Let h_1 and h_2 be any two classifiers. Let C_1 and C_2 be two cost matrices corresponding to loss functions L_1 and L_2. The two cost matrices C_1 and C_2 are "equivalent" ($C_1 \equiv C_2$) iff, for any two classifiers h_1 and h_2 , $L_1(h_1) > L_1(h_2)$ iff $L_2(h_1) > L_2(h_2)$, and $L_1(h_1) = L_1(h_2)$ iff $L_2(h_1) = L_2(h_2)$.*

Theorem 2.1: Let C_1 be an arbitrary cost matrix. If $C_2 = C_1 + \Delta$, where Δ is a matrix of the form

$$\Delta = \begin{vmatrix} \delta_1 & \delta_2 & \dots & \delta_n \\ \delta_1 & \delta_2 & & \delta_n \\ . & & & \\ . & & & \\ \delta_1 & \delta_2 & \dots & \delta_n \end{vmatrix}$$

then $C_1 \equiv C_2$.

For a complete proof of Theorem 2.1 the reader is referred to page 12 of [37]. Transformation of a benefit matrix to a cost matrix according to Theorem 2.1 is shown in Example 2.1. The idea behind such a transformation is to consider benefits of correct classification as lost opportunities in the case of incorrect classifications and add them to the cost of misclassifications. So, in the cost matrix, the incorrect classification entries are sum of resultant costs and lost opportunity values.

Example 2.1: A given benefit matrix B = $\begin{vmatrix} 10 & -3 & -10 \\ -3 & 20 & -20 \\ -15 & -10 & 40 \end{vmatrix}$ can be transformed into an equivalent cost matrix C by adding a matrix which consists of negation of benefit elements

$$\begin{vmatrix} 10 & -3 & -10 \\ -3 & 20 & -20 \\ -15 & -10 & 40 \end{vmatrix} + \begin{vmatrix} -10 & -20 & -40 \\ -10 & -20 & -40 \\ -10 & -20 & -40 \end{vmatrix} = \begin{vmatrix} 0 & -23 & -50 \\ -13 & 0 & -60 \\ -25 & -30 & 0 \end{vmatrix}$$

According to Margeniantu, this transformation does not alter the optimal decisions made [37]. This is true when the base classification algorithm uses only the Equation 2.4 when determining the class of the instance. However, in our algorithm, which is fully dependent on the concept of benefits, and other techniques that incorporate the matrix information inside the core of the algorithm, alteration may occur in the decision process.

2.5 Feature-dependent Benefits

Cost and benefit matrices discussed so far assume that there is a uniform loss or gain value for each kind of classification. To be more precise, the matrices are static and for each instance, classification algorithm uses the predefined entry of the given matrix, independent of the instance itself.

However, in some real-world domains, benefits and costs can be dependent on individual examples, therefore values in benefit and cost matrices may not be constant. For example, consider the credit application domain. When a customer does not repay the loan money he is granted, the bank loses the entire credit amount. On the other hand, if the bank refuses a good customer who is likely to pay the money back, the interest amount that is proportional to the credit loaned will be lost. This situation can be illustrated with the benefit matrix shown in Table 2.5. Here “approve” means to grant the credit loan amount and “deny” means to reject the customer’s request for loan. The term *f(x)* in benefit table denotes the credit amount requested by customer *x*. Obviously, in such a situation, bank officials should be more careful with the high amount requests,

because losses and gains will be much higher. For example, when a customer's request for $10000 is approved and he has defaulted, the benefit of the bank is -$10000, whereas in another application of the same case, if the loan amount is $100, the loss will be much lower, i.e., -$100.

Table 2.5: Benefit matrix for a credit application domain where benefits are dependent on individual instances

	Actual class	
Prediction	approve	deny
approve	$0.5f(x)$	$-f(x)$
deny	$-0.5f(x)$	0

2.5.1 Possible Domains for Feature Dependency

Below is a categorization of domains where benefits can be feature-dependent.

- Financial Domains: As described above, in loan applications, benefits can be a function of the amount queried. In fraud detection of transaction problems, benefits are functions of transaction magnitudes. Moreover, in bankruptcy datasets, benefits might be represented as the size of the bank in dollars. Donation amount prediction as in KDD'98 Cup is another example domain for instance-dependent benefit amounts [5].

- Medical Diagnosis Domains: Benefits of classification can be based on the age of the patients. The younger the patient, the more effective a medication can be in some circumstances. Additionally, there may be temporal parameters associated with patient's health from which benefit functions can be estimated.

- Temporal Domains: In domains where benefits of decisions change over time, it would be more appropriate to specify *f(x)*'s in the benefit table as functions of time. For example, in geo-scientific predictions, like predicting earthquakes, natural disasters, time of prediction is a vital component and benefit of prediction

mostly depends on this parameter. The earlier the prediction is, the more precautions can be taken.

- Spatial Domains: Benefits can be represented as measures of distance in domains where the locality of prediction is important. In weather predictions for example, the rainfall area accuracy is important, and can be a functional parameter for benefit degree.

In this thesis, we have analyzed an example domain, which is bank-loans domain, in which benefits can be dependent on feature values of individual instances. We present a naive approach that is incorporated into the feature projection method.

Chapter 3

Approaches to Cost-Sensitive Learning

Being a recent research area, cost-sensitive learning studies are at their infancy level, and there is plenty of room for improvement in this principal topic. Although preliminary studies were made as early as 1984 by Breiman et al. [11], most of the classification algorithms continue to ignore the asymmetric cost constraints of many real-world situations. Within the last five years, attention over this area has augmented significantly, leading to an online bibliography [3] and a special workshop organization totally dedicated to cost-sensitive learning to be held in 2000 at Stanford University [4]. Recently, fundamentals of the subject are being depicted by Elkan [20] and Turney [46].

In the framework of cost-sensitive learning, costs have been divided into many categories and there is a variety of algorithms working on different cost types. When talking in terms of misclassification costs, there are two major groups of approach. First type of algorithms relies on manipulating the training data whereas the second type studies on converting an error-based classifier into a cost-sensitive one by changing its internal discipline. Margeniantu argues that there is a third approach which manipulates the outputs of the algorithm by probability estimates [37], but we consider such methods

in the second main group. In this chapter, after presenting an overview of these two approaches, studies over feature-dependency of costs are summarized.

3.1 Cost-Sensitive Algorithms that Manipulate the Training Data

Stratification, meta-learning techniques such as MetaCost [16] and boosting are among efforts that manipulate the training data, rather than integrating cost information to the internal classifier.

3.1.1 Stratification Methods

Depending on misclassification cost priorities, predicting a certain class accurately may be more important than predicting the others. If the "important" class is more frequent in the training data, then a standard error-based algorithm is likely to be successful in reducing the total loss, since it will try to minimize errors caused mostly by misclassifying the dominant class. Keeping this aspect in mind, machine learning community has examined the ways to employ an existing error-based algorithm to proper distributions of data such that cost-sensitivity is accomplished. For this reason, some of the researches try altering probability distributions of the original data and build cost-sensitive models using the modified data.

Stratification is the process of changing the frequency of classes in training data in proportion to their cost [16]. There are two methods of stratification, namely *undersampling* and *oversampling*. In undersampling procedure, all examples belonging to the important class are preserved and a fraction of examples belonging to each other class i is chosen at random for inclusion in the reconstructed training set. Although this approach is widely used, it reduces the size of the data available for training, and this may reduce the efficiency of the algorithm while increasing the total cost acquired.

Another alternative method of stratification is oversampling. In oversampling, all examples of class whose erroneous classification is less costly are retained and examples of other classes are duplicated in proportion to their cost values. While doing this, no data is lost, but redundancy in data is increased together with total learning time.

All of the stratification methods distort the original distribution of the dataset. Therefore, classification models learned over stratified datasets do not reflect the reality and many interesting traits may go undetected. In order to overcome these flaws, Chan et al. [12] have proposed a variation of stratification. They have formulated a procedure to convert a natural class distribution to subsets of desired class distributions by replicating the minority class. Then, they apply an arbitrary learning algorithm to each of formed subsets. By the help of a meta-learning strategy such as class-combiner, predictions of the base classifiers are combined.

Chan et al. [12] have tested their approach only in a single domain, namely credit card fraud detection. They have observed that, the training class distribution have larger effects on cost performance than cost-based sampling or stratification. However, they confess that there is an unavoidable need to run preliminary experiments to determine the desired class distribution which is highly dependent on the cost model.

3.1.2 Boosting Methods

Instead of modifying the class distributions, some techniques deal with changing the weights of instances provided to the algorithm. This weight adjustment should be processed in such a way that new weights reflect the impact of cost distribution. Boosting is a multi-classifier approach that operates with this initiative. It is a general method of iteratively enhancing the performance of a classifier by the help of an instance reweighting methodology [50]. Boosting forms new models based on strengthening the old models' weak points and combines all decisions made by those classification models by a voting scheme. A fundamental algorithm on boosting is AdaBoost which is recently being studied and extended [2].

Main idea of AdaBoost is to form multiple individual classification models in sequential runs and to adjust the weights of training instances so as to maximize the performance [50]. It begins with assigning equal weights to all instances in the training data. Then, it calls the learning algorithm to form a classifier for this data and reweights each instance according to the correctness of the classifier's decisions. The weight of a misclassified instance is increased effectively so as to make its classification more important in the next iteration. Respectively, the weight of a correctly classified instance is decreased. These adjusted weights cause the base learner to concentrate on different examples in each turn. After a finite number of generations which build models on reweighted data, individual classifiers are combined by means of a voting procedure [42].

There are several recent attempts to make AdaBoost cost-sensitive in the literature. The natural way of doing this is to use the cost of misclassifications to update the training data weights on successive boosting rounds. One of such variations is presented by Fan et al. in [22]. They have integrated a misclassification cost adjustment function into the weight updating formula of AdaBoost. This function increases the weights of more costly instances while decreasing the weights of inexpensive examples relatively. Their method is mostly applicable to situations where misclassification costs are relatively stable. They have evaluated their algorithm by comparing it with original AdaBoost procedure and the results show that AdaCost is superior to AdaBoost in reducing misclassification costs.

Two other cost-sensitive variants of boosting have been proposed by Ting et al. in [45]. Their study differs from AdaCost in a way that methods are based upon tree classifications in the situation where misclassification costs change very often. In their first approach, the minimum expected cost criterion is used to select the predicted class. They have used a modified version of C4.5 decision tree algorithm [41], i.e., C4.5c as the base learner. During classification stage, at the leaf of the tree, C4.5c calculates the expected misclassification cost for every class and chooses the predicted class with the lowest expected cost for a given instance.

The second approach of Ting et. al in [45], which is called cost-boosting, entirely modifies the weight updating rule of AdaBoost. According to new rule, if an instance is

misclassified, its weight is replaced with its misclassification cost; otherwise its weight remains unchanged. Their reported results have shown that cost-boosting is a better approach for reducing costs than simple boosting with minimum expected cost criterion.

In his further studies, Ting improved his boosting approaches by presenting two new variants [43]. All these alternatives should relearn their models when misclassification cost information changes. For evaluation of the effectiveness, he has compared four boosting methods, namely CSB0, CSB1, CSB2 and AdaCost. In the result of experimentation, the mean relative cost is reduced by a small margin, i.e., less than 10%, for first three variants and is increased by 5% for AdaCost. Ting also points out the deficiencies in AdaCost weight updating procedure and shows directions for improving it [43].

3.1.3 Meta-learning Methods

Some approaches to cost-sensitive learning treat the internal base classifier as a black box and wrap a meta-learning stage around that base in order to tune it in presence of fluctuating costs. MetaCost [16] is one of such meta-learning methods and it has become a benchmark for comparison between cost-sensitive classification algorithms.

MetaCost, as originally defined by Domingos, relies on a bagging algorithm. It firstly starts by forming multiple bootstrap replicates of the training set and learning a classifier on each. Then, by using the votes of this ensemble of classifiers, it tries to estimate the probability of each class for a given instance. Using these approximated probabilities, MetaCost algorithm relabels each training instance with the estimated optimal class and then reiterate the classifier on the relabeled training set. The pseudo-code for MetaCost algorithm is given in Figure 3.1.

Input: S is the training set,

L is a classification learning algorithm,

C is a cost matrix,

m is the number of resamples to generate,

n is the number of examples in each resample,

p is *True* iff L produces class probabilities,

q is True iff all resamples are to be used for each example.

Procedure MetaCost (*S, L, C, m, n, p, q*)

For i = 1 to m

Let S_i be a resample of S with n examples.

Let M_i = Model produced by applying L to S_i.

For each example x in S

For each class j

Let $P(j \mid x) = \frac{1}{\sum_i 1} \sum_i P(j \mid x, M_i)$

Where

If p then $P(j \mid x, M_i)$ is produced by M_i

Else $P(j \mid x, M_i) = 1$ for the class predicted

by Mi for x, and 0 for all others.

If q then i ranges over all M_i

Else i ranges over all M_i such that x $\notin S_i$.

Let x's class = argmin$_i$ $\sum_j P(j \mid x) C(i, j)$

Let M = Model produced by applying L to S.

Return M .

Figure 3.1: The MetaCost Algorithm [16]

One difference of MetaCost from Chan et al.'s method [12] is that it does not have to repeat all the runs when the cost matrix changes. Only the final learning stage is needed to be rerun, and thus making MetaCost more flexible to variations in the cost matrix.

Another advantage of MetaCost is its generic form and ability to introduce cost-sensitivity aspects to any error-based classifier.

MetaCost has been shown to outperform undersampling and oversampling stratification methods, and reduced cost compared to C4.5 error-based classifier. However, Ting [43] argues that Domingos made no comparison between MetaCost's final model and the internal cost-sensitive bagging model. When MetaCost is compared to a cost-sensitive bagging or boosting method, Ting has showed that the latter algorithms give better results and thus, meta-learning stage of MetaCost burdens more computation than necessary. His study suggests that a classifier with cost-sensitive elements may outperform a generic cost-sensitive wrapper method like MetaCost applied to an error-based classifier. So, it is more beneficial to directly incorporate cost information to the classifier itself.

Another wrapper approach is studied by Lin et al.[35]. Their method initially uses a logistic model to minimize number of misclassification errors, then uses a cost sensitive algorithm which is a variant of Breiman's bagging [10] and MetaCost [16]. It takes into account not only the misclassification costs but also the prior probabilities. Their target domain is prediction of financial distress. In the result of their observations, Lin et al. claim that cost sensitive learning should also consider the prior probabilities whenever possible.

Weka [6], which is a famous implementation platform of machine learning algorithms, has implemented a meta-cost-sensitive classifier which uses two methods to introduce cost factors to its base classifier. First method is to reweight training instances according to the total cost assigned to each class, and second method is to directly predict the class with the minimum expected misclassification cost. The second method requires the base classifier to be distribution classifier, which outputs the estimated probabilities of classes for instances.

3.2 Algorithms That Are Modified To Be Cost Sensitive

There have been various attempts to make different classifiers sensitive to misclassification costs. Most of these studies have focused on decision trees whereas there is number of studies over decision lists, naïve Bayesian classifiers and case-based reasoning, a.k.a. CBR, systems. In addition, there is a direct attempt of using estimated probability outputs in minimization of total misclassification costs.

3.2.1 Decision Trees

The earliest efforts to incorporate variable misclassification costs into the process of decision tree induction were made by Breiman et al. In [11], two different methods adapting the test selection criterion in the growing stage of the tree are described. One of these methods was reported to infer negative results by Pazzani et al.'s empirical study [39] Their observation was that cost-sensitive trees do not always have lower misclassification than the conventional error-based trees. The naïve approach of using error costs as test selection metric is investigated in [39]. For this purpose, the partitions of the training set made by each possible test are found initially. Then the test that minimizes the sum of costs of all partitions is selected. However, this approach did not produce desired results when compared to standard decision tree metrics, mostly due to the problem of overfitting.

Contrary to pre-processing approaches, Webb proposes a post-processing strategy to lower costs [47] His strategy is inspired by the theorem of decreasing inductive power. This theorem suggests that elements of a classifier having high misclassification costs should be specialized so as to minimize the proportion of false positives to true positives. In terms of decision trees, elements to be specialized are leaves of the tree. In this strategy, as leaves associated with classes of high costs are specialized, leaves having lower costs are generalized respectively. Webb presents a theoretical analysis of this concept together with its application to C4.5 decision tree inducer. In order to achieve

this goal, C4.5CS which is a decision tree post-processor is employed and he has reported a slight reduction in misclassification costs. He also notes that the effect of specialization is smaller for pruned trees than unpruned ones. One interesting aspect of specialization approach is that it does not need accurate misclassification costs, the only need is the relative ordering of them. However, in such a case, how the accurate degree of agreement between specialization and the cost model will be determined, is an open question.

In contrast to Pazzani et al's study, Ting claims that, a truly cost-sensitive tree can be learned directly from the training data [44]. For this purpose, the greedy divide and conquer algorithm is coalesced with a simple heuristic. Specifically, weights of the instances that are modified proportionally to the cost of misclassifications are used in place of the number of instances in the standard greedy divide-and-conquer. They have converted C4.5 to C4.5CS (same naming for the second time in literature) by employing this methodology and their approach seeks to minimize the number of high cost errors, rather than minimizing the total misclassification cost. An interesting note made by Ting [43] at this point is that, minimizing the number of high error costs does not guarantee to achieve minimization in the total misclassification cost. This is because as the algorithm avoids high cost errors, the number of consequential low cost errors is usually increased. Margineantu in [37] has investigated ways to manipulate weights in order to incorporate general cost matrices into decision tree algorithms as well. He presents a general wrapper method and five other techniques for determining weights for growing decision trees.

Bradford et al. have studied decision tree pruning for minimizing loss together with probability estimation techniques [8]. They have extended existing pruning methods to involve cost-complexity characteristics and formed two variants of pruning based on Laplace corrections. Results obtained in their studies indicate that no method dominates the others in all datasets and furthermore, different pruning mechanisms are better for different cost matrices. They also show that Laplace correction performs well compared to others, for some cost matrices.

Another study dealing with pruning methods of decision trees is [17]. Drummond et al. have investigated the effects of the splitting criteria and pruning methods over

expected misclassification costs. They have shown that decision tree splitting criteria in common use are relatively insensitive to costs and class distribution. Two methods have been suggested in their study [17]; one is completely treating decision tree with cost-insensitive splitting and pruning techniques and the other is to grow decision tree cost-independently and then prune it in accordance with the costs. Second approach intersects greatly with Webb's [47] specialization.

Zubek et al. have also scrutinized the effects of pruning the search space for the sake of cost minimization [52]. They have considered misclassification costs together with attribute measurement costs, i.e., test costs. Their algorithm is based on formulating the classification process as a Markov Decision Process. Zubek et al.'s admissible search heuristic is shown to reduce the problem search space remarkably. In addition, to reduce overfitting, they have introduced a supplementary pruning heuristic named "statistical pruning".

3.2.2 Decision Lists

Pazzani et al. have studied two algorithms concerning decision lists, first is called Reduced Cost Ordering for creating decision lists and second one is the Clause Prefix method for avoiding overfitting in decision lists [39]. Reduced Cost Ordering algorithm firstly initializes the decision list to a default rule that guesses the least expected cost class. Then, by replacing the default rule with a new rule, it tries to progress upon the available decision list. This strategy results in significantly lower costs than Reduced Error Ordering, which tries to minimize the error rate and most of the time better than the decision tree approaches studied in [39].

Clause Prefix method [39] is a pruning algorithm which is designed to be used in combination with Reduced Cost Ordering algorithm. It is based on finding all prefixes of each clause that is learned and adding them to the pool of clauses from which Reduced Cost Ordering algorithm selects clauses that have more prediction power in less literals. However, similar to the case of decision trees, this pruning method is shown to have no significant effect over minimizing the cost.

3.2.3 Naive Bayesian Classification

Cost-sensitivity issue has also been examined in the context of other classification algorithms such as Naive Bayesian Classification. Pazzani et al. have also studied cost-sensitive decision making with Bayes classifier among their decision tree approaches [39]. Bayes-Cost simply assigns an instance to the least expected cost class which is determined by function of the probability estimates returned by the classifier. Empirical results show that Bayes-Cost does well when the data does not violate the independence assumption and there are few irrelevant features, otherwise it performs poorer.

In [27], Gama et al. have presented an iterative approach to naive Bayes which also exhibits cost-sensitive properties. This approach consists of building distribution tables by naïve Bayesian techniques at first, and then applying an optimization process. The optimization process is based on an iterative update of the contingency tables and it aims to improve the probability class distribution associated with each training example. When there are non-uniform error costs in the domain, this iterative update can be guided by misclassification costs and, in such a situation, contingency tables are updated according to correct or incorrect classifications made. Experimental results over UCI benchmark datasets show that this method brings advantages over error-based and stratification based naive Bayesian classification in most of the datasets.

3.2.4 CBR Systems

Cost-sensitive CBR systems have been investigated by Wilke et al.[49]. KNN_{cost} which is a modified version of KNN algorithm is presented in order to learn feature weights for classification improvement of CBR systems. Their method is based on conjugate gradient and it uses an integrated decision value matrix within the error function. They have shown that their method based on cost minimization is much more effective than their method based on accuracy, namely KNN_{acc} and both provide improvements over initial CBR systems. However, their evaluation has only covered one application domain which is credit scoring domain of very limited size, and they have not compared their algorithm

to other existing methods. For that reason, we cannot fully decide on the predictive power of their approach.

3.2.5 Direct Cost-sensitive decision making

Zadrozny et al. have proposed a method called direct cost-sensitive decision making [51]. This study is based on the idea that any learned classifier that can provide conditional probability estimates for training data can also estimate conditional probabilities for test data of the same domain. By means of those estimated probabilities, Zadrozny et al. claim that the optimal prediction labels of test examples can directly be computed. By testing their approach using five different probability estimation methods over the KDD'98 dataset, they have reported better results than MetaCost, which uses the same probability estimation methods on C4.5 with pruning and collapsing. This result is not surprising, since it has also been approved by [43] that MetaCost usually does not perform better than an internal cost-sensitive classifier.

3.3 Approaches to Feature-Dependent Misclassification Costs

In the literature of cost-sensitive learning, there are few studies which have included feature-dependent aspects of cost matrices. As mentioned in section 2.5.1, in several real-world domains, prediction outcomes may be dependent on specific feature values that vary for different instances. In such a case, although misclassification type is the same, costs can be considerably diverse.

Fawcett et al. are among the first ones who incorporated feature-dependent costs in their classification problem. In cellular cloning fraud detection [23] used a variable cost matrix based on the fraudulent airtime used. Naturally, this is due to the more cost of prolonged fake calls. Static cost notion is inappropriate in such situations.

Since the credit card fraud detection domain is extremely dependent on the dollar amount of each credit card transaction, Chan et al. in their studies [12] and [13] represented the cost model in terms of overheads, which are equivalent to operational costs that is needed for each investigation and transaction amounts of instances. If the amount of transaction is smaller than the overhead, net gain will be lower even if that transaction is fraudulent, so it is not worthwhile to make an investigation. They have examined the effects of cost-based sampling, which samples instances in proportion to their transaction amount ratios, but their concluded performance is not much different from random sampling. Instead, as stated in Section 3.1.1, they pointed out that variations in training set class distributions have more promising effects on cost performance.

Recently, Hollmen et al. [26] have examined feature-dependency, i.e., input dependency concept thoroughly and pointed out that there is an indisputable area of applications in which cost matrices of functions should be used instead of fixed cost matrices. They present a cost model and decision function based on Bayesian formulation. Posterior probabilities they make use of are obtained by a Hidden Markov model. The observed variables are assumed to be conditionally dependent on a discrete hidden variable in the HMM structure. Their input-dependent cost model exhibits promising results in terms of profit performance, when compared to cost-neutral and fixed-cost approaches in fraud detection of telecommunications domain. Furthermore, it is stated that the described cost model is applicable with other methods such as neural networks or probabilistic networks. However, Hollmen et al. make a footnote that this approach is favorable when the input-dependent cost model is easily formulated [26].

Elkan [19] take one step further and ask the question “What will happen if instance-dependent costs $C(i,j,x)$ are unknown for some labels i and j, for some training examples x?” This question is worthwhile to be considered with great attention. Such situations occur when costs are functions of features that are dependent on the class label, such as charity donation amounts, and practically impossible to be known beforehand. In [51], it is further emphasized that estimating unknown costs can be more important than estimating probabilities. The method Zadronzy et al. use to predict instance-dependent cost amounts is least-squares multiple linear regression. By looking at the examples in the

training set, costs for test instances are predicted. Simple methods are used for probability and cost estimations and their study tries to provide an insight and a baseline for future research. More sophisticated regression methods for cost estimation are likely to give more satisfactory improvements in this research area [51].

Chapter 4

Benefit Maximization with Feature Intervals

The concept of benefit, i.e., the worth of correct classification, has been undervalued in the literature of cost-sensitive learning. Most of the cost-sensitive algorithms presume that correct classifications have no further interpretations other than simply conveying no cost to the domain. However, as described in Chapter 2, there may be non-uniform distributions of benefits between accurate predictions in particular domains. Bearing this aspect in mind, we have investigated the benefit maximization problem where there are different merits related to different class labels.

In this study, we have chosen to integrate the notion of benefit maximization into the framework of feature projections technique. The most important advantages of this feature projection representation are its flexibility of organization, robustness to noisy training instances and missing feature values together with the resultant short training and classification time. Additionally, in the end of the prediction process, it produces a human readable model of the classification knowledge.

In this chapter, we firstly describe feature projection concept and introduce a new cost-sensitive feature projection based learning algorithm, namely Benefit Maximization with Feature Intervals (a.k.a. BMFI). Subsequently, training and querying with BMFI is discussed comprehensively, along with time and space complexity analysis of the methods.

4.1 Knowledge Representation

Feature projections for knowledge representation constitute the background for BMFI algorithm. Feature projection technique is another exemplar-based learning methodology in which intervals formed on features are independent units of knowledge. BMFI inherits the knowledge representation scheme of feature projection technique of early FIL, VFI, VFI5 and CFI algorithms ([14],[28],[29] and [30]).

4.1.1 Feature Projections Concept

In a particular classification problem, given the training dataset consisting of m features, an instance x can be thought as a point in an m-dimensional space with an associated class label x_c. It is represented as a vector of nominal or linear feature values and its associated class label, i.e., $<x_1,x_2,..,x_m,x_c>$. Here, x_i represents the value of the ith feature of the instance x. If we consider each feature separately, and take x's projection onto each feature dimension, then we can represent x by the combination of its feature projections. Figure 4.1 illustrates such a situation in a three-dimensional space.

BMFI algorithm first projects all training instances on each feature separately. As a result, over each linear dimension of features, instances are marked as points in the beginning. Using these projections, BMFI algorithm constructs a set of *intervals* for each feature. An *interval* can either be a *point* or a *range* interval. Point interval stands for a single feature value, whereas range interval is a set of consecutive values of the particular feature.

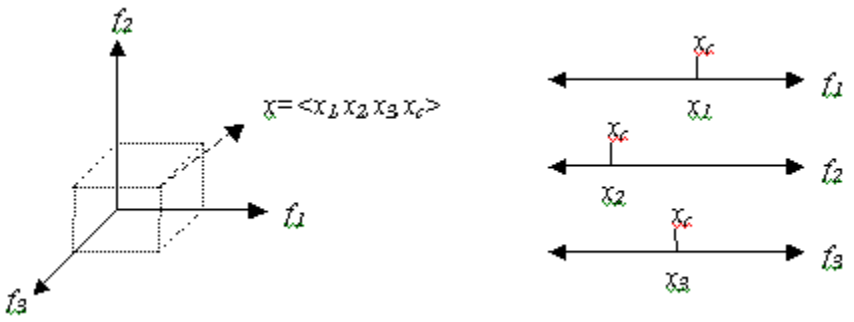


Figure 4.1: A simple feature projection illustration for a single instance

Definition 4.1: *An interval on a feature f is represented by a vector of the following format:*

$$I = < lb, ub, N_1, N_2, \ldots, N_k, V_1, V_2, \ldots, V_k >$$

where k is the number of classes in the domain, N_j is the number of instances belonging to class j in and V_j represents the vote of the interval I for class j. The first two elements of the vector *lb* and *ub* denote the lower and upper bound values of the interval, respectively.

Definition 4.2: *A point interval is an interval* $I = < lb, ub, N_1, N_2, \ldots, N_k, V_1, V_2, \ldots, V_k >$ *such that* $lb = ub$ *on feature f.*

As the name implies, point intervals are single-valued projections of features. It is noteworthy that a nominal feature's projection dimension consists of point intervals only.

Definition 4.3: *A range interval is an interval* $I = < lb, ub, N_1, N_2, \ldots, N_k, V_1, V_2, \ldots, V_k >$ *such that* $lb < ub$ *on feature f.*

In the training phase, the feature intervals of concept definitions are constructed by generalization and specialization [31]. When forming intervals, those instances that have the same value on a feature dimension are grouped into point intervals. By looking at neighboring point intervals that share the similar characteristics are combined to form

range intervals. Merging of intervals for generalization purposes is mentioned in detail in the training phase of the algorithm in section 4.2.

An example training set and the corresponding feature interval construction steps are shown in Figure 4.2. The example domain consists of three features, namely f_1, f_2 and f_3, the first two of which are linear and the last one is a nominal feature. The nominal feature f_3 can take values from the set $\{A,B,C\}$. The class labels are C_1, C_2 and C_3. There are seven training instances in the example.

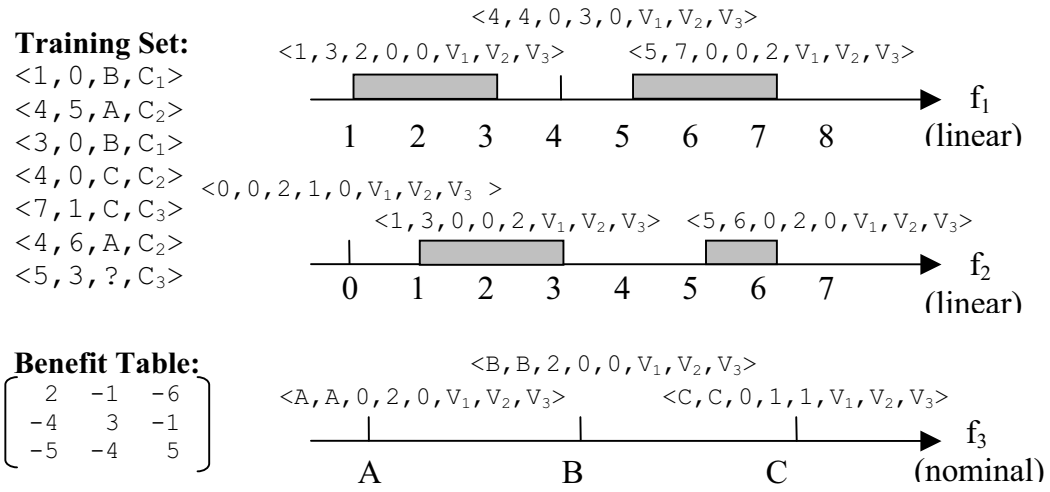


Figure 4.2: Example demonstrating the formation of feature intervals

Training algorithm forms three intervals on the feature f_1, two of which are range intervals. The first interval on f_1, spans the value range [1,3], and there are only 2 instances of C_1 in that interval. Vote assignment will be discussed later on in this chapter.

4.1.2 Basic Notions for Benefit Maximization on Feature Intervals

Definition 4.4: *A voting method, VM for short, is a function of the form f(I)→g(I) that takes interval I as an input and assign votes to classes on that interval by a predefined routine.*

For example, most of the error-based classifiers generate probability estimation outputs when predicting an unseen example. In such a circumstance, a possible voting method for a feature projection based classifier might be assigning votes according to the number of instances belonging to each class in the particular interval. Specifically, votes can be set according to the formula:

$Vote_I(c)$ = Number of instances of class c in interval I

Another voting method may compute ratios of instance occurrences in the interval such that

$Vote_I(c)$ = (number of instances of class c in I) / (total number of instances in I)

These probability estimations on the interval structure are analogous to decision tree leaf structure. In benefit maximization, these voting methods should consider benefits or costs of the classifications before casting votes to intervals. One possible approach is simply using expected benefits, computed according to Equation 2.5, as vote values. Different versions of BMFI with a number of voting methods are introduced in Section 4.2.

Definition 4.5*: Benefit of classifying a single instance x of class k as class c is denoted as*

$$B(x,c) = \begin{cases} B[c,c] & ,c = k \\ B[c,k] & ,c \neq k \end{cases}$$

given the benefit matrix B.

Definition 4.6: *A class c is the beneficial class of an interval I iff for* $\forall j \in C$ *and* $j \neq c$ $\sum_{x \in i} B(x,c) \geq \sum_{x \in i} B(x,j)$*, where C is the set of all possible classes.*

Beneficial class concept solely relies on the interval structure. Considering the interval as an independent and distinct unit of knowledge, it is questioned whether it would be worthwhile to predict all the instances as of a certain class label. Thus, if the prediction problem has a simple benefit matrix such as

	Actual Class	
Prediction	Class 0	Class 1
Class 0	1	-2
Class 1	-1	3

and the interval I has a signature $I=<0,40,10,5,V_1,V_2>$ then interval I's beneficial class will be class 1, even though there are six more instances of class 0 than instances of class 1. The subsequent calculations verify this fact:

$$\sum_{x \in i} B(x,0) = (10 \times 1) + (5 \times -2) = 0$$

$$\sum_{x \in i} B(x,1) = (5 \times 3) + (10 \times -1) = 5$$

Therefore, regardless of other factors, it will be more advantageous to predict all instances as class 1, rather than predicting all as class 0, in this interval. From a cost-sensitive point of view, it is often more sensible to predict the most beneficial class instead of the most probable class.

Definition 4.7: *Minimum benefit, denoted as minB(S), of an instance set S, is the sum of B[j,k]'s of predictions $j \in C$ such that $\forall x \in S$ having true class k, $j \neq k$ and B[j,k] is minimal.*

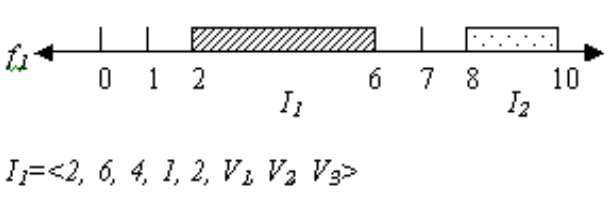
$$minB(S) = \sum_{x \in S} \arg\min_{j \neq k}(B[j,k])$$

Here, instance set S can either be a test instances set or an interval's instance set. When S is an interval's instance set, minimum benefit is the worst benefit that might take place when all the predictions made by the interval are incorrect. For example consider the benefit matrix B supplied and the interval formation shown in Figure 4.3.

In interval I_1, there are four instances of C_1, one instance of C_2 and two instances of C_3. The least beneficial classification for an instance of C_1 is misclassifying it as C_3, which has a benefit of -8. Conversely, C_2 instances lead to the smallest amount of benefit

if classified as C_3. Prediction row of C_3 shows that the worst case classification is accompanied with a benefit of -12 in $B[C_1,C_3]$. Thus, the lower bound of benefit obtainable for the situation in Figure 4.3 is calculated as follows:

$$minB(I_1)= \sum_{x\in i_1}\sum_{j\neq k} \arg\min(B[j,k]) = (4\times-8)+(1\times-10)+(2\times-12) = -66$$



$B =$

	C_1	C_2	C_3
C_1	10	-3	-8
C_2	-5	20	-10
C_3	-12	-7	15

Figure 4.3: An example interval formation

Definition 4.8: *Maximum benefit, denoted as maxB(S), of an instance set S, is the sum of $B[x_c,x_c]$'s for $\forall x \in S$, such that true class of x is x_c.*

$$maxB(S) = \sum_{x\in S} B[x_C, x_C]$$

Maximum benefit of an interval is the highest benefit obtainable when all the instances of that interval are correctly classified. Of course, Definition 4.8 relies on the assumption that all benefit matrices in consideration obey the reasonableness conditions. In a similar fashion to the example above, if we consider the circumstances shown in Figure 4.3, the maximum benefit obtainable by using B is

$$maxB(i_1) = (4\times10)+(1\times20)+(2\times15) = 90$$

Identification of possible upper and lower bounds of accessible benefits are crucial for determining the benefit accuracy of the algorithm. With respect to those limits, a benefit scale is formed for evaluating the efficiency of the algorithm. Details of this performance scale are given in Section 5.1.

Definition 4.9: *Confidence of an interval I is the difference between the highest vote and the second highest vote available in I.*

Definition 4.10: *Benefit confidence of an interval I is the difference between benefits of the most beneficial class and second beneficial class of I.*

Both confidence values indicate the prediction strength and assertion of intervals. If confidence of an interval is low, i.e., near zero, then the decision made by the interval is somewhat tentative. Confidence tests are used for finding and coalescing uncertain intervals to form more robust ones.

4.2 Training with BMFI

The training process of BMFI algorithm is shown in Figure 4.4. In the beginning, for each feature *f*, all training instances are sorted with respect to their value for *f*. This sort operation is identical to forming the projections of training instances for each feature *f*. A point interval is constructed for each projection. Initially, the lower and upper bounds of the interval are equal to the *f* value of the corresponding training instance. If the *f* value of a training instance is unknown, it is simply ignored. Then, if there are several point intervals as the same *f* value, then they are combined into one point interval by adding the class counts. At the end of point interval construction, vote for each class label is assigned by a predefined voting method. This voting method is mostly based on benefit matrix provided as an additional input to the algorithm. Five of the possible voting methods are described in detail in subsection 4.2.1.

After determination of votes on point intervals, consecutive point intervals are scanned for combination only for linear features. This generalization operation can be done by means of several issues, such as frequency of class labels, majority of votes or benefit of the intervals. Details of generalization are presented in subsection 4.2.2. After combining intervals to form more general ones, some of them can be pruned as an optional routine. Pruning is useful when the classification model created becomes so much dependent on the nature of the training set. This problem is referred as *overfitting* [38] and should be avoided as much as possible. In addition to voting and generalization

methods, different pruning approaches are studied and discussed in detail in subsection 4.2.3.

```
train(TrainingSet,BenefitMatrix)
begin
    for each feature f
      sort(f, TrainingSet)          /* sort TrainingSet with respect to f */
      /* construct a list of point intervals using feature values and class labels */
    interval_list ← make_point_intervals (f, TrainingSet)
    for each interval i in interval_list
        /* cast a vote for each class in the interval using the instances in interval */
        vote_i(c) ← voting_method (i,f,BenefitMatrix)
    if f is linear
        /* join adjacent point intervals to form range intervals */
        interval_list ← generalize (interval_list,BenefitMatrix)
        if (pruning=yes)
            interval_list ← prune (interval_list,BenefitMatrix)
end.
```

Figure 4.4: Pseudo-code of the training stage in BMFI algorithm

4.2.1 Voting Methods of BMFI

As explained above, votes on the class labels in intervals can be determined by considering several different factors. There are two main approaches to make voting methodology sensitive to cost factors. First approach adjusts probability estimations to reflect the importance of beneficial predictions and second approach directly incorporates benefit matrix knowledge to the voting. Here, we introduce and discuss several methods of voting and their possible outcomes.

4.2.1.1 Probabilistic Voting

This type of voting is profitable when the importance of the class is proportional to its distribution in the training dataset. As discussed in Chapter 3, there are various wrapper algorithms that try to make its internal error-based classifier sensitive to misclassification costs by altering the class distribution ratios. Without altering these ratios, if class distribution knowledge is included in the algorithm, then more reliable predictions which exhibit the natural characteristics of domain can be attained. However, there is a major handicap in determining a general approach when using only probabilistic methods, i.e., the classification importance of a class can be directly or indirectly proportional to its frequency in the dataset. Assuming an optimistic world-view, most of the fraudulent, illegitimate or hazardous cases rarely occur and in such situations, classification of the rare class is more important. On the other hand, in a medical domain, it is more likely to encounter poor health incidents more often than healthy cases, since people usually see a doctor only when they are sick. In such domains, a traditional error-based classifier which tries to minimize number of errors made rather than total misclassification cost is likely to perform well. Nevertheless, probability estimation methods that favor the prediction of the rare classes are less likely to achieve desired success. This observation reinforces the fact "There is no universally best classifier" [24].

Since conventional error-based learning algorithms predicts the most frequent class to reduce the number of errors, we look at the situations where rare classes are more important and introduce probabilistic approaches that are suitable to such domains.

Voting Method 1 (VM1):

The first scheme that we will investigate is the voting method of the original CFI algorithm [28], called *VM1* in our context, which can be formulated as follows:

$$VM1(c, I) = \frac{N_c}{classCount(c)} \tag{4.1}$$

where N_c is the number of instances that belongs to class c in interval I and *classCount(c)* is the total number of instances of class c in the entire training set. Note that this sort of voting does not take benefit of classifications into account. Consider the interval $I_k = <2,10,15,10,V_1,V_2>$ in which 15 instances belong to C_1 and 10 instances belong to C_2. In the whole training set, suppose there are 100 instances of C_1 and 50 instances of C_2. According to Equation 4.1, V_1=15/100=0.15 and V_2=10/50=0.20. As it can be seen from these results, although C_1 is more frequent in the interval, its vote can be lesser than the other class because the second class is rarer in the whole dataset. By this voting method, classification of a single rare class instance becomes more important in proportion to its scarcity in the training set.

When testing the performance of algorithmic approaches, VM1 produced surprisingly well results, mostly due to its simplicity and natural characteristic of favoring rare classes. However, it ignores the benefit information and this deficiency makes it still far from optimal benefit accuracy when used on its own.

Voting Method 2 (VM2):

VM2 is the second probabilistic voting method and is a slight modification of VM1. When assigning votes, it simply uses the ratio of interval class distribution to the dataset class distribution. More formally, VM2 is formulated as:

$$VM2(c,I) = \frac{(N_c / total(I))}{(classCount(c) / No\_Trainers)} \tag{4.2}$$

where N_c is the number of instances that belong to class c in interval I, *total(I)* is the total number of instances falling into interval I, *classCount(c)* is the total number of instances of class c in the entire training set and as the name implies *No_Trainers* is the total number of training instances. The ratio in the denominator of Equation 4.2 is sometimes called *base rate* of class c [51].

VM2 is more complex than VM1 and it is questionable whether it will suffer or benefit from this complexity in terms of benefit performance issues. In Chapter 5, the answer to this question is examined.

4.2.1.2 Beneficial Voting

This type of voting is more general than probabilistic voting since it is dependent on benefit matrix information rather than class distributions. In this approach, the benefit matrix is directly embedded in the voting scheme and that's why we call it *beneficial voting*.

Beneficial voting is applicable to both situations where the rare or the frequent class prediction is more important. Its prediction power can be supported by accurate probability estimations to gain higher benefit performance. Here, we will consider three different variations of beneficial voting.

Voting Method 3 (VM3):

The first method of beneficial voting determines the votes of classes in an interval by using the relative benefit of the classes. The vote of class *c* is computed by

$$VM3(c,I) = \frac{total\_benefit(c,I) - minB(I)}{maxB(I) - minB(I)} \tag{4.3}$$

where *minB(I)* and *maxB(I)* are the minimum and maximum benefit possible in interval *I* and *total_benefit(c,I)* is the benefit of labeling all instances in the interval as class c. More formally,

$$total\_benefit(c,I) = \sum_{x \in I} B(x,c) \tag{4.4}$$

Note that $B(x,c)$ is the benefit of labeling instance x as class c.

This voting scheme is similar to benefit accuracy calculation that is used in evaluating the performance of the algorithm. VM3 normalizes the votes of the interval, i.e., maps

them to [0,1] range, by subtracting the minimum possible benefit in the interval and then by taking the resultant's ratio to the possible maximal range of benefits in the interval. Evidently, this voting method makes direct use of the benefit matrix entries and makes decisions based on importance of the classes.

Voting Method 4 (VM4):

An example of beneficial voting supported by probabilities is *VM4*. VM4 is based on VM3 with an additional multiplicand of interval class probability. It can be formulated as

$$VM4(c,I) = \frac{total\_benefit(c,I) - minB(I)}{maxB(I) - minB(I)} \times \frac{N_c}{total(I)} \tag{4.5}$$

where the first multiplicand of the right-hand side of Equation 4.5 is same as VM3 formula, N_c is the number of instances that belongs to class c in interval I and *total(I)* is the total number of instances falling into interval I.

Voting Method 5 (VM5):

The last but not the least voting method that we will introduce in this chapter is *expected benefit voting,* a.k.a. *VM5*. This voting mechanism is totally founded on optimal prediction approximation given in Equation 2.5. Thus, VM5 casts votes to class c in interval I by

$$VM5(c,I) = \sum_{k \in C} B[c,k] \times P(k \mid I) \tag{4.5}$$

Here, B[c,k] is the benefit matrix entry that represents the benefit of predicting an instance of class k as class c and $P(k \mid I)$ is the estimated probability that an instance falling to interval I will have the true class k. There are various methods for estimating probabilities and some of them have been explored above. However, by the empirical results we obtained, we decided to use the following probability estimation:

$$P(k \mid I) = \frac{N_k}{classCount(k)}$$

Again, here N_k is the number of instances that belongs to class k in interval I and *classCount(k)* is the total number of instances of class k in the entire training set.

There are several other methods we have tried and eliminated during the empirical evaluation of classifiers. In Chapter 5, a throughout comparison between different types of voting on various datasets is presented.

4.2.2 Feature-dependent Voting

All of the methods described above assume that there is a static benefit associated with each pair of predicted and actual class labels. However, there may be situations where the benefit matrix is defined in functional dependency forms. When the benefit matrix of classification is dependent on features, votes of intervals should be arranged in a way to reflect this feature dependency.

In feature projection concept, each feature is assumed to be an independent unit of knowledge and they individually contribute to voting with equal prediction power. However in feature-dependent conditions, the dependent variable would have some effect on decision of other feature dimensions as well. For this reason, it is not very straightforward to incorporate feature dependency concerns to an autonomous environment like feature projections. In this thesis, we handle feature dependency on the dependent variable's dimension only.

Pseudo-code in Figure 4.5 summarizes the routine followed in feature-dependent voting scheme. It uses VM5 as the base voting method. Other beneficial voting methods can be employed as well, but expected benefit calculation is more suitable to our current implementation.

```
/* Input: f is the feature, interval list is the corresponding intervals on dimension f. */
assign_dependent_votes (interval_list, f )
begin
for each i in interval_list do
    if f is a dependent variable
        /* take the average value dependent in the interval */
        avg ← average(i.upper , i.lower)
        vote_i(c) ← VM5(i,f,BenefitMatrix)×avg
    else
        vote_i(c) ← VM5(i,f,BenefitMatrix)
end.
```

Figure 4.5: Pseudo-code for assigning feature-dependent votes

Consider a medical diagnosis domain, in which the benefit of classification depends on the age of the patient. In this yes-no classification problem, an example matrix can have the following format:

		Actual Class	
	Prediction	healthy	ill
	healthy	f_{age}	$-2\,f_{age}$
	ill	$-1\,f_{age}$	$3f_{age}$

where f_{age} represents the age feature in the dataset. This benefit matrix implies that accurate diagnosis in elder patients is more important, because side effects caused by wrong classification may be more damaging (In some cases, this situation can be vice versa and accurate prediction of younger patient may be more important.). On f_{age} dimension, votes will be arranged as follows. For $I_1=<30,40,20,7,V_h,V_i>$ $B_h = (20+(7\times-2))\times35 = 180$ and $B_i = (-20+(7\times3))\times35 = 35$. For $I_2=<45,60,15,20,V_h,V_i>$ $B_h = (15+(20\times-2))\times52.5 = -1312.5$ and $B_i = (-15+(20\times3))\times52.5= 2362.5$. These values should be normalized by subtracting the minimum benefit of interval and mapped to the

[0..1] range by dividing into possible range of benefits. Taking the average of interval feature value is necessary, since it would be extremely costly and redundant in terms of space to hold each instance's value. So, we take the average and find the effective value through the interval.

4.2.3 Generalization of Intervals

After votes are assigned to each interval with respect to some specified criteria as described, point intervals should be generalized to form range intervals. This process can be likened to clustering process of machine learning. Successive intervals showing similar characteristics are joined to form more general and informative units of knowledge. There is a number of ways to carry out this combination process. One group of policies looks at the votes of the interval for combination, and the other group considers the possible benefits of each consecutive interval.

The generic form of generalization process is illustrated in Figure 4.6. The *merge_condition*() function is a comparison mechanism that evaluate relative properties of each interval and returns true if sufficient similarity level between those intervals is reached. Some join operations combine two consecutive intervals, whereas some operations compare and combine three consecutive intervals to form a larger single interval.

Besides adding more prediction power to the algorithm, proper generalization of intervals reduces the number of intervals to a great extent, and thus, decreases the classification time substantially. However, careful attention must be paid when combining intervals, because minor yet vital information related to rare occurrences of instances may go undetected when excessive generalization take place. In this section, some of such generalization policies are examined and exemplified. Subsequently in the next chapter, their effects to classification progress are investigated in depth.

```
generalize (interval_list)
begin
    I← first interval in interval_list
      while I is not empty do
          I' is the interval after I
          I" is the interval after I'
          if merge_condition(I,I',I") is true
          then merge I'(or I") into I
          else I←I'
end.
```

Figure 4.6: Pseudo-code for generalization of intervals

4.2.3.1 Joining Intervals That Have the Same Frequent Class (SF)

First method to combine two consecutive intervals is to test whether their most frequent class is the same. The intervals whose highest class counts are for the same class are merged to form range intervals. So, in this process, *merge_condition_SF*() is defined as

```
merge_condition_SF (I,I')
    if (frequent_class(I) = frequent_class(I'))
        then return(true)
    else
        return(false)
```

where *frequent_class*(*I*) function returns the class label that have the highest class count in the interval *I*.

An example demonstrating this join process is shown in Figure 4.7. Here, on projection dimension of f_1, which is a linear feature taking values between 0 and 100, there are five point intervals initially. The specified domain has two classes to predict and instances of the training set are projected onto f_1 dimension as shown on interval signatures. Instance counts of point intervals imply that the first three consecutive intervals have the same frequent class, i.e., C_1. In the first execution of the while loop in the generalize function given in Figure 4.6, I_1 and I_2 are joined to form I_1'. When joining

intervals, number of instances belonging to each class is summed up and their corresponding votes are rearranged. In the second step, seeing that I_1' and I_3 have the same majority class, they are combined to form I_1''.

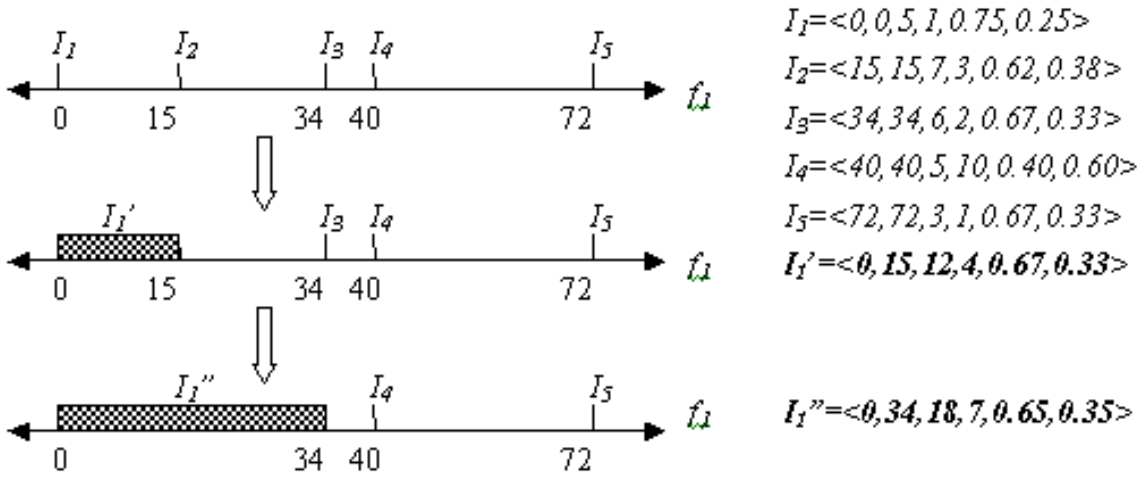


Figure 4.7: An example demonstrating merge operation of same frequent class intervals

In the example above, number of intervals is reduced from five to three. During the classification progress these resultant three intervals will be used in search and vote operations, so there will be a noticeable reduction in classification time.

4.2.3.2 Joining Intervals That Have the Same Beneficial Class (SBC)

A second methodology for generalizing intervals is to join consecutive intervals that have the same beneficial class. This approach relies strictly on benefit matrix provided to the algorithm. If the beneficial classes of two consecutive intervals are the same, then it can be more profitable to unite them into a single interval. Corresponding pseudocode for SBC is as follows:

```
merge_condition_SBC(I,I')
        if (beneficial_class(I) = beneficial_class(I'))
                then return(true)
        else
                return(false)
```

To exemplify this process, let us consider the following benefit matrix in a binominal problem dataset

		Actual class	
Prediction		C_0	C_1
C_0		1	-3
C_1		-2	4

and the two consecutive intervals I_1 and I_2 having signatures I_1=<lb_1,ub_1,10,15,V_1,V_2> and I_2=<lb_2,ub_2,5,5,V_1,V_2> respectively. The beneficial class of I_1 is C_1, calculated as follows:

$$\sum_{x \in I_1} B(x,0) = (10 \times 1) + (15 \times -3) = -5$$

$$\sum_{x \in I_1} B(x,1) = (15 \times 4) + (10 \times -2) = 40$$

The beneficial class of I_2 is also C_1 and these two intervals can be joined to form a new and more general interval I_1'=<lb_1,ub_2,15,20,V_1,V_2>.

$$\sum_{x \in I_2} B(x,0) = (10 \times 1) + (5 \times -3) = -5$$

$$\sum_{x \in I_2} B(x,1) = (5 \times 4) + (10 \times -2) = 0$$

4.2.3.3 Joining Intervals That Have High Confidence Values (HC)

This type of generalization looks at three consecutive intervals and joins them into a single interval. By this type of generalization, intervals that are formed from redundant or noisy data are eliminated and their information is assimilated to the other two. Generalization takes place if *merge_condition_HC*() is satisfied :

merge_condition_HC(*I,I',I''*)
if (*majority_vote(I) = majority_vote(I'')*) and
(*confidence(I) > confidence(I')*) and
(*confidence(I'') > confidence(I')*)
then return(*true*)
else return(*false*)

The illustration of this process is given in Figure 4.8. In this example there are four intervals with their specified signatures. Problem domain consists of three class labels to predict and according to the differences between those votes, the intervals are decided to be joined or not.

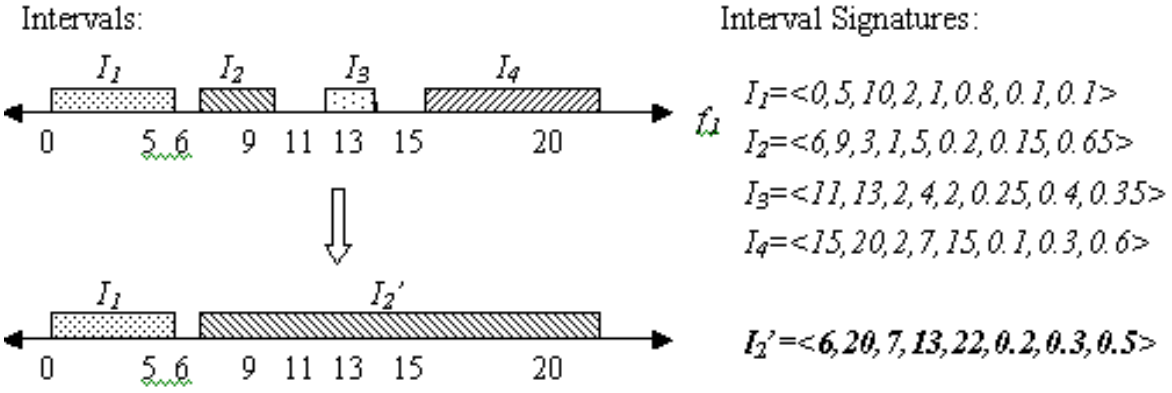


Figure 4.8: Example for illustrating merging high confidence intervals

As seen, I_1 has the majority class C_1, whereas I_2 and I_4 have majority class C_3 and I_2 has majority class C_2. The loop in the generic *generalize*() function checks intervals in a three-wise fashion. In first iteration, it checks whether I_1 and I_3 have the same majority class. In our case, this is not the situation, so no combination takes place we iterate to the second trio, which is I_2, I_3 and I_4. Seeing that I_2 and I_4 share the common majority class, we check their confidences. Here I_2 has a confidence of 0.45 and I_4 has a confidence of 0.3. These values are much higher than the confidence of the middle interval I_3's confidence 0.05. So, according to our join operation, we combine those three intervals into a single one, by adding the instance counts and recalculating class votes.

4.2.3.4 Joining Intervals That Have High Benefit Confidences (HBC)

Joining high benefit confidence intervals has the same logic as joining high confidence intervals with the exception that all calculations are based on benefits rather than majority of the votes. The process again combines three consecutive intervals to a single one, if the middle interval has less prediction power then the other two. Corresponding formulation of condition when such three intervals are to be merged is as follows:

merge_condition_hb(I_{left},I_{middle},I_{right})
if (*beneficial_class*(I_{left}) = *beneficial_class*(I_{right})) and
(*benefit_confidence*(I_{left}) > *benefit_confidence*(I_{middle})) and
(*benefit_confidence*(I_{right}) > *benefit_confidence* (I_{middle}))
then return(*true*)
else return(*false*)

Continuing with the example given in Figure 4.8, with regards to benefit matrix

	Actual class		
Prediction	C_0	C_1	C_2
C_0	1	-1	-3
C_1	-1	2	-4
C_2	-2	-2	4

benefit and benefit confidence of each interval is calculated as follows: For I_1, $\sum_{x\in I_1} B(x,0) = 5$, $\sum_{x\in I_1} B(x,1) = -10$ and $\sum_{x\in I_1} B(x,2) = -20$. So, for I_1, the beneficial class is C_0 with a benefit confidence of 15. For I_2, $\sum_{x\in I_2} B(x,1) = -13$, $\sum_{x\in I_2} B(x,2) = -21$ and $\sum_{x\in I_2} B(x,2) = 14$. These calculations indicate that I_2 has a benefit confidence of 27 for beneficial class C_2. For I_3, corresponding benefit confidence is 2 favoring the beneficial class of C_1. In I_4, beneficial class is C_2 with a relatively high confidence of 90. Hence, according to these values, I_2 and I_4 have the same beneficial class and their benefit confidences are much higher than the middle interval I_3. Thus, although the beneficial class of the middle interval is different, those three intervals are joined into a single one,

because the middle interval I_3 is not certain about making a decision in terms of benefit. Again, by this merge operation, number of intervals has been reduced from four to two.

4.2.4 Benefit Maximizing Pruning of Intervals

Pruning, as in the case of decision trees is a powerful tool for avoiding the problem of overfitting and reducing the problem size. Hence, we decided to include pruning techniques in our BMFI algorithm and investigated its effects over the benefit performance. In decision tree discussions, advantages of pruning techniques are still being questioned [39] and a consensus on this issue has not been reached yet. Bearing this issue in mind, we have investigated effects of pruning on benefits in the context of feature projection based classification.

In this section, we will introduce a single method for eliminating redundant or disadvantageous feature intervals and in Chapter 5 we will discuss effects of this procedure on various datasets. Overall, choice of pruning seems to be dependent strictly on the nature of the domain and the amount of redundancy available in the training datasets provided.

Our approach to interval pruning is based on the comparison of the votes of the most beneficial class and majority voted class. If the vote of the most beneficial class is less than the highest vote to some extent, i.e., beneficial class is undervalued in the interval, then pruning that interval may be a good choice. The reason behind this is usually in such intervals there is a more evenly distribution of classes and thus, that feature value range is somewhat uncertain about making prediction. Elimination of these dangling intervals may be profitable since such an operation will shift the power of voting to more “certain” intervals. Corresponding pseudo-code of this procedure is given in Figure 4.9.

In the code of Figure 4.9, *major* is the class that receives the highest vote from interval *I* and *most* is the most rewarding class when predicted. After the determination of these classes in a particular interval, the standard deviation between all the votes of *I* is calculated. If the difference between the votes of majority class and beneficial class

exceeds this standard deviation, then *I* is pruned. This standard deviation test allows us to check whether votes of the interval are in accordance with the benefit concerns. If there is critical gap between, then it is better to simply ignore that interval.

```
prune(interval_list)
begin
    for each I in interval_list
        most ← beneficial class of I
        major ← majority class of I
        std ← standard_deviation(I.votes)
        if ( vote[major] – vote[most] ) is greater than std
            prune I
end
```

Figure 4.9: Pseudo-code of prune operation on intervals

4.3 Classification with BMFI

After the prediction model is constructed on the training data, it is time to classify previously unseen data. The classification (querying) process of the BMFI algorithm is given in Figure 4.10. BMFI classification stage is very similar to that of CFI [28] and it involves a voting scheme where each feature acts as an independent unit and casts its vote for the particular instance's class.

The process starts by initializing the votes of each class label to zero. If the value of the query instance q for a feature f, i.e., q_f is unknown (missing), then that feature does not involve in the voting process. Rather than altering the characteristics of the instance (i.e., by assigning average quantities for unknown feature values), simply ignoring that feature dimension is a more natural and straightforward way of handling missing values. If q_f is known, then the interval I into which q_f falls is searched. If the q_f does not fall in any interval of f, then again, the feature f does not participate in the voting. If an interval I is found that covers the q_f value, then the votes of that particular interval are the votes it

casts in the overall voting operation. Once all features have completed casting their votes, the class that received the highest amount of the votes is predicted as the class of the query instance q.

```
classify (q) /* q: query instance to be classified */
begin
    /* initialize total votes */
    for each class c
        v_c ← 0
    /* go over each feature dimension and sum up votes */
    for each feature f
      if q_f is known
        I ← search_interval(f, q_f)
       for each class c
        v_c ← v_c + interval_vote( I, c)
      /* predicted class is the one with the maximum votes */
    prediction ← arg max_c (v_c)
return prediction
end.
```

Figure 4.10: Classification phase of BMFI

It should be noted that in our classification scheme, each feature has an equal power in the voting. This condition is guaranteed by normalizing the votes of each interval in the training phase of BMFI. Several policies that play with the weights of features in voting can be employed, especially when costs of features are asymmetrical. However, those policies are beyond our current discussion and are subject to future research.

When querying examples using a constant benefit table, the total benefit is calculated by simply adding up the corresponding benefit matrix entry for each test instance. In that case, each type of classification, e.g. classifying i as j, has identical revenue. On the other hand, if a feature dependent benefit table is available in the domain, for each query example there is a different benefit gained and it is the functional form of feature values

identified in the table. For each instance in the test set, these functional measures are summed up to calculate the total resultant benefit.

4.4 Time and Space Complexities of BMFI

When evaluating performance of a classification algorithm, learning time and disk storage needed are important concerns along with the predictive accuracy. An algorithm having a superior accuracy is not favorable and practical if it runs in exponential time and/or needs excessive amounts of disk space. Our BMFI algorithm runs in a considerably well amount of time and has manageable space requirements. In this section of the thesis, we will investigate estimations of these figures in terms of algorithmic variables. In Table 4.1, the list of those algorithmic variables is given.

Table 4.1: Input variables of BMFI

N	number of training instances
f	number of features in the domain
l	number of linear features in the domain
k	number of possible classes
r	maximum number of distinct values of a feature
t	number of test instances

4.4.1 Time Complexity of BMFI

Training:

With reference to training pseudo-code re-presented in Figure 4.11, all the training instances are processed *f* times to form point intervals on each feature dimension *f*. Hence, *make_point_intervals*() function requires O(*N*) time for a single feature pass. As a result of this pass, for *r* distinct feature values, *r* point intervals are formed initially. Calculating votes of these intervals requires constant time and total operation on *r* intervals takes O(*r*) time. Generalization of intervals makes a second pass over the interval list of *l* features, that consists of at most *r* intervals. If pruning is employed, then it is processed over at

most *r* intervals again, assuming the worst case that no joining on intervals took place. Therefore, pruning on the worst case consumes O(*r*) for a single feature's interval list.

```
train(TrainingSet,BenefitMatrix)
begin
    for each feature f                                                          }
      /* sort TrainingSet with respect to f */                                  }
      sort(f, TrainingSet)                                                      }
      interval_list ← make_point_intervals (f, TrainingSet)   O(N)              }
    for each interval i in interval_list                      O(r)              }
          vote_i(c) ← voting_method (i,f,BenefitMatrix)       O(k)              } O(fN)
      if f is linear                                                            }
      /* join adjacent point intervals to form range intervals */               }
      interval_list ← generalize (interval_list,BenefitMatrix)   O(l)           }
      if (pruning=yes)                                                          }
         interval_list ← prune (interval_list,BenefitMatrix)     O(l)           }
end.
```

Figure 4.11: Runtime evaluation of training phase of BMFI

Overall, each of interval formation, generalization and pruning operations are carried out for each feature in the domain, and at worst case, run time of BMFI is $O(f\times(r+N))=O(fr+fN)$, since $f \geq l$. When each feature have a distinct value on a particular feature *f*, *r* can be at most *N*, and therefore, at the worst case, runtime of the algorithm is $O(2f\times N)=O(f\times N)$ which is a quadratic and very efficient training time.

Classification (Querying):

In the classification process given in Figure 4.10, classification of a single instance requires a pass over all feature values of that instance. For each *f*, a binary search over *f*'s intervals is performed and this requires O(log *r*) time on the average with *r* intervals. At the worst case, when no generalization and pruning is carried out and there are *N* distinct values for feature *f*, then *r*=*N* and worst case search requires O(log*N*) time. For *f* features,

the total time is O(f×log$N$). Votes for k classes are summed up for each feature entailing O(f×k) time. So, overall time needed for classification of a single instance is O(f×log$N$+f×k)=O(f×log$N$), since log$N$ >> k. As a result, the classification of total test data is done in O(t×f×log$N$) time. On the average case, runtime of the classification is much lower than this upper bound since there will be a substantial reduce in the number of intervals by generalization and pruning operations.

4.4.2 Space Complexity of BMFI

When training BMFI, there can be at most N distinct values and thus N intervals for a feature f. So, we need a total of f×N space to hold interval structure. Interval structure itself consists of 2k +2 values, i.e., k for holding number of instances belonging to each class, k for holding their corresponding votes and 2 for holding lower and upper bound values of the interval. This equals to O(k). Therefore, storage requirement is O (f×N×k) initially. As the algorithm progresses, there will be extensive shrink in this upper limit due to the reduction in number of intervals.

Chapter 5

Experimental Results

When dealing with cost-sensitivity, class imbalance in the datasets is the main issue faced with. Importance and consequences of this problem have been widely addressed in [48]. Most of the error-based algorithms fail when the minority class is more valuable in the domain, and in some contexts, cost-sensitive classification has become the process of detecting minority class. However, a generic method which is applicable to all sorts of domains is needed. That's why we think that benefit information should directly be used.

In this chapter, we will investigate the behavior of BMFI algorithm in various domains. Consequences of employing different voting methods to different class distributions are monitored, together with the effects of generalization of intervals and pruning over these voting methods. Later on, comparisons of BMFI with wrapper cost-sensitive algorithms using Naïve Bayesian Classification and C4.5 decision tree learner are presented.

5.1 Benefit Accuracy

In order to evaluate the efficiency of cost-sensitive algorithms, there are a few proposed methods, such as Receiver Operating Characteristics (ROCs), area under ROC curves

(AUCs), average cost per instance calculations [37]. ROC curve evaluation has the flexibility to make comparisons between classifiers when the cost matrices are not strictly defined. However, it has its own drawbacks and a more simple representation that reflects the precision of classification in terms of benefit is needed. For this reason, we propose an accuracy metric which is defined as follows:

Definition 5.1*: Benefit accuracy of a classification model M in domain D over the instance set S is the normalized ratio of gained benefit to the maximum possible benefit, i.e.,*

$$\text{Benefit Accuracy}_{\text{D}}(M,S) = \frac{\text{Benefit}_{\text{D}}(M,S) - \text{minB}_{\text{D}}(S)}{\text{maxB}_{\text{D}}(S) - \text{minB}_{\text{D}}(S)}$$

Here, $\text{Benefit}_D(M,S)$ is the benefit obtained by model M on domain D, $\text{minB}_D(S)$ and $\text{maxB}_D(S)$ is the minimum and maximum benefit obtainable in domain D respectively. That is, $\text{minB}_D(S)$ is the total benefit achievable when all the test instances are classified as the worst wrong case. Similarly, $\text{maxB}_D(S)$ is obtained when all instances in S are classified correctly. When $\text{Benefit}_D(M,S)$ is equal to the minimum benefit possible, then benefit accuracy of the model is 0. Correspondingly, when it equals the maximum benefit possible, then benefit accuracy is 1, as expected. In other words, this metric maps the obtained benefit to [0..1] range as in the case of conventional predictive accuracy. To be more specific, benefit accuracy is the general form of classical predictive accuracy. When the diagonal elements of the benefit matrix are one and non-diagonal elements are all zero, i.e., all types of classifications have equal importance and there is no cost for misclassifications, then benefit accuracy equals predictive accuracy used in comparison of error-based classifiers.

It should be noted that benefit accuracy metric not only compares relative benefits of two classifiers, but also indicates the algorithm's efficiency over the particular domain. For this reason, we have chosen to evaluate our algorithm mainly with regards to benefit accuracy. All the accuracy results presented in this chapter are the average benefit accuracies achieved when 10-fold cross validation is utilized over the entire datasets. That is, for each dataset, initially the instance space is partitioned into 10 equal-sized

subsets. The algorithms are run 10 times using a different subset each time as the test set and the remaining nine as the training set. By this process, it is guaranteed that the training sets are disjoint and each instance in the whole dataset is classified exactly once. Resultant benefit accuracy is the average of the accuracy values of these 10 runs. Gained total benefit, $\text{Benefit}_D(M,S)$, is another metric that is used to evaluate cost-sensitivity.

5.2 Datasets and Benefit Matrices

For evaluation purposes, we have used several benchmark datasets taken from UCI ML Repository [7] and datasets that we constructed from real-world domains. Individual characteristics of the datasets influence results obtained significantly, since cost-sensitivity is extremely correlated to class distributions. Thus, we first give fundamental information about the datasets used throughout the experimentation stage. Later on, details of benefit matrix construction are presented.

5.2.1 Properties of Datasets Used

Datasets that we have used in evaluation of our algorithm can be divided into three groups. First group of datasets consists of five binary datasets taken from UCI ML repository. In these datasets, one class is assumed to be more important to predict correctly than the other by a constant benefit ratio. Basic properties of these two-class datasets are given in Table 5.1. In this table, base rate represents the ratio of the important class instances in the dataset. These five two-class datasets are chosen to characterize each possible sort of distribution, e.g. in breast-cancer and diabetes datasets, the important class is the minority class, whereas in ionosphere and liver disorders datasets the important one is the majority class. Sonar dataset is chosen because there is a natural near-equal distribution of instances among class labels.

A detailed explanation of these datasets is provided in Appendix A. In two-class benchmark datasets, the benefit matrix is not explicitly specified and thus, we have chosen to set the positive class as the most important one in classification. For medical

datasets, this is the existence of the particular disease, i.e., classification as being ill is considered as more important.

Table 5.1: Basic properties of two-class benchmark datasets from UCI ML Repository

Dataset	**# of instances**	**# of features**	**# of linear features**	**# of classes**	**Base Rate**
Breast-cancer-w	699	9	9	2	0.3447
Pima-diabetes	768	8	8	2	0.3490
Ionosphere	351	34	34	2	0.6410
Liver disorders	345	6	6	2	0.5797
Sonar	208	60	60	2	0.5336

Table 5.2: Basic properties of multi-class benchmark datasets from UCI ML Repository

Dataset	**# of instances**	**# of features**	**# of linear features**	**# of classes**	**Class Distributions**	
Ecoli	336	8	8	8	C_1=143 C_2=77 C_3=52 C_4=35	C_5=20 C_6=5 C_7=2 C_8=2
Glass	214	9	9	6	C_1=70 C_2=76 C_3=17	C_4=13 C_5=9 C_6=29
Page-blocks	5473	34	34	5	C_1=4913 C_2=329 C_3=28	C_4=88 C_5=115
Vehicle	846	18	18	4	C_1=199 C_2=217	C_3=218 C_4=212
Wine	178	13	13	3	C_1=59 C_2=71	C_3=48

The second group of datasets consists of multi-class datasets, again taken from UCI ML repository [7]. Table 5.2 lists basic characteristics of these datasets together with class distributions. These datasets are chosen among the ones that have an abundant number of linear features, in order to evaluate the effectiveness of generalization and pruning methodologies. As it can be observed, these five multi-class datasets are selected to reflect properties of various possibilities and different class distributions. We have assigned two types of benefit matrices to these datasets, for evaluating cost-sensitivity issues. One of the matrices is constructed so as to facilitate the prediction of minority

classes. Construction of such matrices is explained in the next subsection. We have also used a random benefit matrix which gives random importance levels to prediction of different classes. Both of the matrices that have been used in experiments and the details of these datasets are given in Appendix A.

Other than benchmark datasets, there are five datasets that we have used in our studies. Fundamental characteristics of these datasets are given in Table 5.3. To evaluate benefit aspects, there are pre-assigned benefit matrices, which have been determined by experts of the specified domains or by us after a careful examination of domain specifications, for these datasets. All these benefit matrices are presented together with detailed information about the datasets, in Appendix B.

Table 5.3: Five special datasets which have their own individual benefit matrices

Dataset	**Size**	**# of features**	**# of linear features**	**# of classes**	**Base Rate**	
Bank-loans	1443	13	7	2	C_1=1000	C_2=443
Bankruptcy	1444	19	19	2	C_1=1030	C_2=414
Dermatology	366	34	33	6	C_1=112 C_2=61 C_3=72	C_4=49 C_5=52 C_6=20
Lesion (Gastric Carcinoma)	285	68	7	9	C_1=3 C_2=55 C_3=7 C_4=103 C_5=6	C_6=6 C_7=17 C_8=69 C_9=19
Arrhythmia2r	526	279	272	2	C_1=245	C_2=281

5.2.2 Benefit Matrix Construction

In our experiments, we constructed two-class benefit matrices by the help of a constant *benefit ratio* which denotes the ratio between benefits of correct classification of each class. For two-class datasets, we used benefit ratio values 2, 5, 10, 20, 50 and benefit matrix of the format:

	Actual Class	
Prediction	C_0	C_1
C_0	1	- *b*
C_1	-1	*b*

where *b* denotes the benefit ratio constant. This matrix can be interpreted as correct prediction of C_1 is *b* times more beneficial than C_0.

For multi-class problems, if the particular dataset does not have a predefined benefit table, one is constructed with respect to the distribution of classes in the dataset. When the minority class is assumed to be more important, construction procedure is done as follows: If the probability distribution of class *i* is represented as *P*(*i*), the benefit of classifying an instance of class *i* as class *j* is assigned as -[*P*(*j*)/*P*(*i*)]×*b* where *b* is a constant representing the importance degree of correct classification of class *j*. Diagonal elements are determined by multiplying the benefit ratio with the inverse of probability of that class occurrence. Constant multiplicand of diagonal elements can be varied in order to adjust the matrix to represent the features of the domain. Below is an example illustrating the construction of a benefit matrix from scratch: If instances of C_1 and C_2 constitute up 0.2 and 0.3 of the dataset respectively, and C_3 makes up the remaining 0.5; an example benefit matrix with a benefit ratio of 5 is constructed as:

	Actual Class		
Prediction	C_1	C_2	C_3
C_1	[1/P(1)] × *b*	- [P(1)/P(2)] × *b*	- [P(1)/P(3)] × *b*
C_2	- [P(2)/P(1)] × *b*	[1/P(2)] × *b*	- [P(2)/P(3)] × *b*
C_3	- [P(3)/P(1)] × *b*	- [P(3)/P(2)] × *b*	[1/P(3)] × *b*

If we place the corresponding P(*i*) values, the resultant benefit matrix *B* is as follows:

		Actual Class		
	Prediction	C_1	C_2	C_3
B =	C_1	25	-3.33	-2
	C_2	-7.5	16.67	-3
	C_3	-12.5	-8.33	10

It can be clearly observed that, constructed benefit matrix adjusts benefits in inverse proportion to the probability distribution of the class. This configuration favors prediction of the rare class. On the contrary, if importance of the class is in direct proportion to its class frequency, then we can form an appropriate benefit matrix by taking the inverse of the ratios in the above construction and adjusting the diagonal elements appropriately.

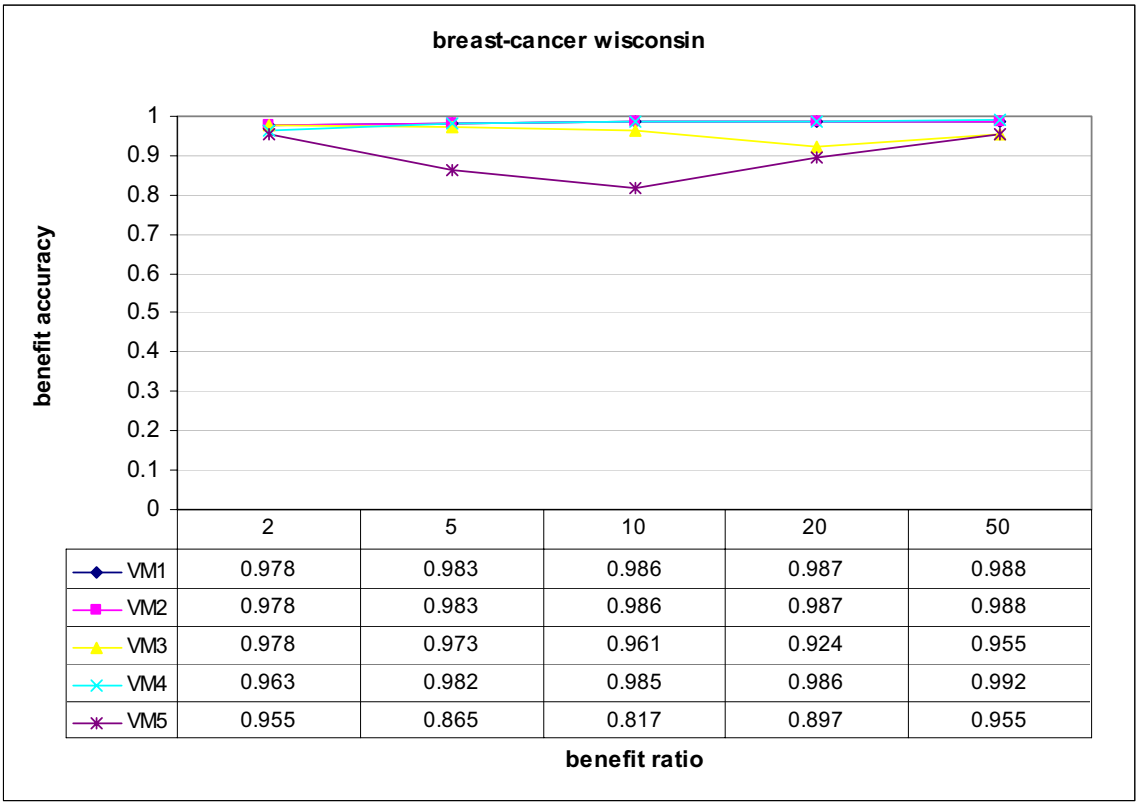
5.3 BMFI Comparisons

In this section, we present an incremental progress of BMFI versions. First, we give comparative test results of five different voting methods on various datasets. Later on, generalization and pruning effects over voting methods are illustrated through a number of experiments. Overall evaluation on combined effect of techniques and voting methods are discussed consequently.

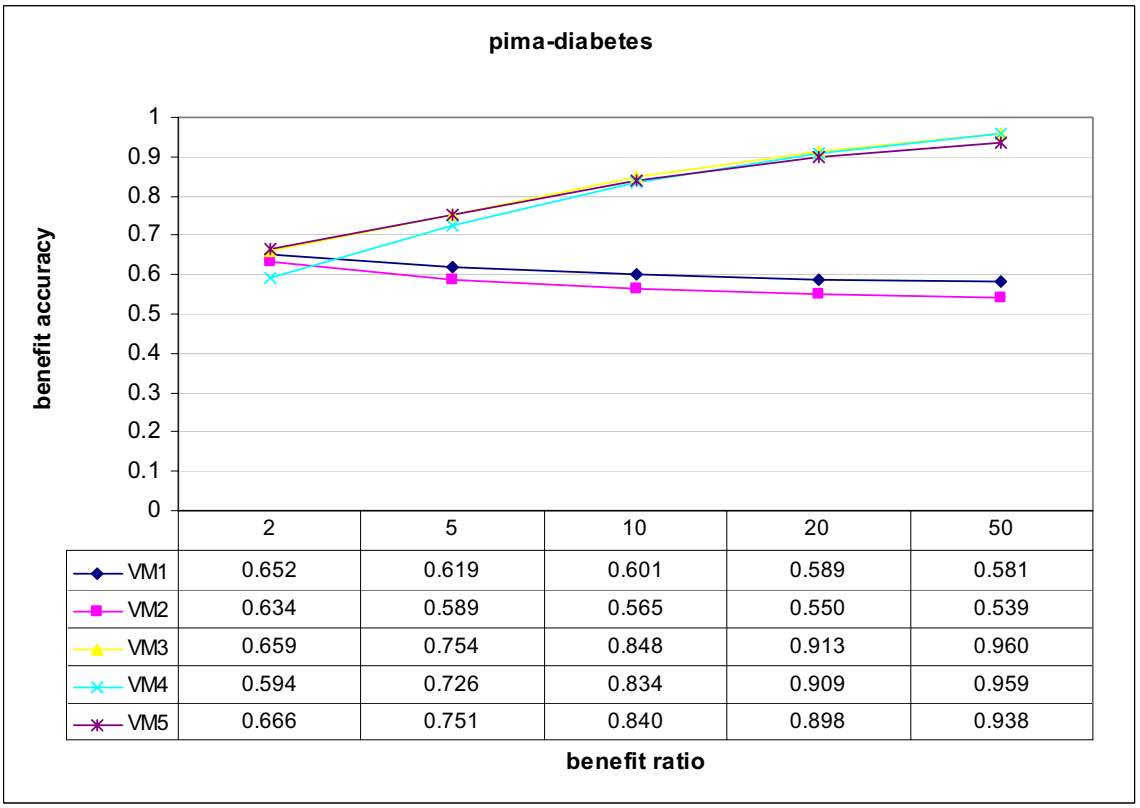
5.3.1 Comparison of Voting Methods

As described in Section 4.2.1, there are five voting methods we have employed during the experimentation stage of BMFI. Two of these voting schemes are probabilistic methods that favor the prediction of rare classes. The remaining three methods use benefit table information to value possible prediction outcomes. In Figure 5.1, the behavior of these five voting methods on two-class datasets and changing benefit ratios are presented.

When a probabilistic voting method is used singly, the classifier becomes almost an error-based classifier that does not consider benefits of classifications (we say “almost” because this type of voting is strictly dependent on class distributions and it favors the prediction of minority class). As the results of Figure 5.1 illustrate, beneficial voting methods outperform error-based approach especially when the benefit ratio is increased. Not surprisingly, when the benefit ratio is low, probabilistic voting logic becomes more adequate in the domain.

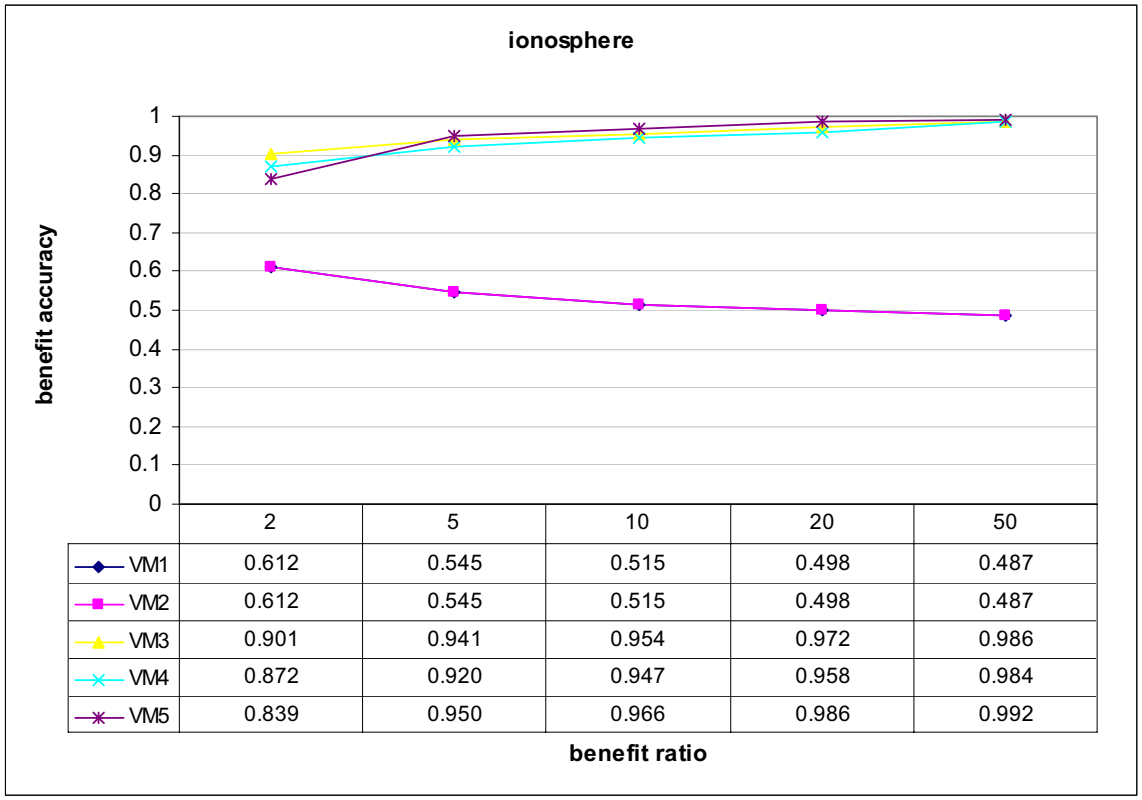


	2	5	10	20	50
VM1	0.978	0.983	0.986	0.987	0.988
VM2	0.978	0.983	0.986	0.987	0.988
VM3	0.978	0.973	0.961	0.924	0.955
VM4	0.963	0.982	0.985	0.986	0.992
VM5	0.955	0.865	0.817	0.897	0.955

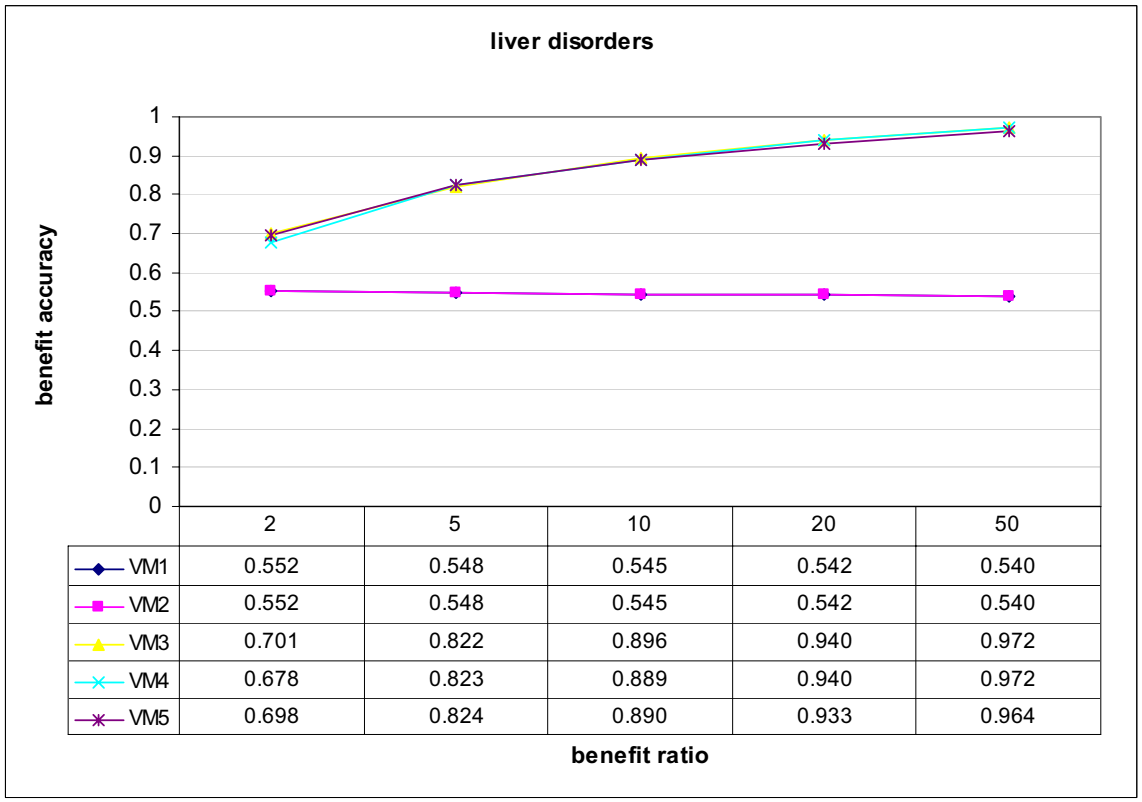


	2	5	10	20	50
VM1	0.652	0.619	0.601	0.589	0.581
VM2	0.634	0.589	0.565	0.550	0.539
VM3	0.659	0.754	0.848	0.913	0.960
VM4	0.594	0.726	0.834	0.909	0.959
VM5	0.666	0.751	0.840	0.898	0.938

Figure 5.1: Behavior of voting methods on two-class benchmark datasets

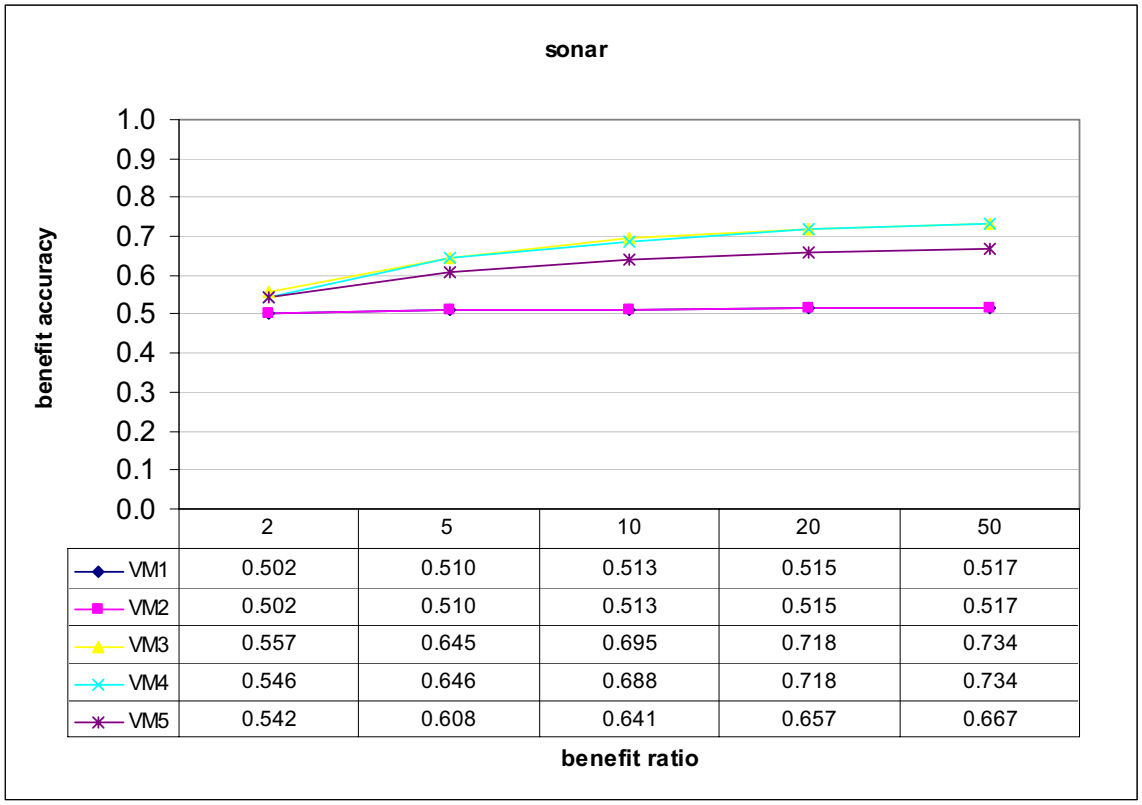


	2	5	10	20	50
VM1	0.612	0.545	0.515	0.498	0.487
VM2	0.612	0.545	0.515	0.498	0.487
VM3	0.901	0.941	0.954	0.972	0.986
VM4	0.872	0.920	0.947	0.958	0.984
VM5	0.839	0.950	0.966	0.986	0.992



	2	5	10	20	50
VM1	0.552	0.548	0.545	0.542	0.540
VM2	0.552	0.548	0.545	0.542	0.540
VM3	0.701	0.822	0.896	0.940	0.972
VM4	0.678	0.823	0.889	0.940	0.972
VM5	0.698	0.824	0.890	0.933	0.964

Figure 5.1(cont.): Behavior of voting methods on two-class benchmark datasets



	2	5	10	20	50
VM1	0.502	0.510	0.513	0.515	0.517
VM2	0.502	0.510	0.513	0.515	0.517
VM3	0.557	0.645	0.695	0.718	0.734
VM4	0.546	0.646	0.688	0.718	0.734
VM5	0.542	0.608	0.641	0.657	0.667

Figure 5.1(cont.): Behavior of voting methods on two-class benchmark datasets

Both of the probabilistic methods VM1 and VM2 performed poorly on two-class datasets except breast-cancer dataset, and beneficial voting methods have acquired considerably well results on each of the datasets. With the increase in benefit ratios, there is an observable proliferation in benefit accuracies when beneficial voting is used. On the other hand, rise in benefit ratios cause a slight reduction in benefit accuracies in the case of probabilistic voting. This reduction is due to the lack of benefit information in probabilistic voting.

On multi-class datasets, behavior of voting methods is presented in Table 5.4. These datasets are evaluated by using a pair of benefit matrices, one favoring the prediction of rare class and the other assigning importance levels randomly. In Table 5.4, the difference of accuracy between series of rare_matrix and random_matrix are natural since they operate on different benefit matrices, i.e., different baselines. What is really important is the local maximum of these series. According to those results, VM1 and VM5 are the most promising voting methods when rare class prediction is more

important. The reason why VM1 become this much efficient in these datasets is that, in the construction of benefit matrices, relatively small benefit ratios, i.e., in the range of 2-4, are used. Conversely, VM3 causes the worst results in four of the five multi-class datasets. When random matrix is used, VM4 produces considerably well results. On the average, however, VM5 is the winner of performance in majority of the multi-class datasets.

Table 5.4: Behavior of voting methods over multi-class datasets

Ecoli

	VM1	VM2	VM3	VM4	VM5
rare_matrix	0.72051	0.626925	0.572779	0.631834	0.611721
random_matrix	0.487577	0.455996	0.434554	0.468542	0.406952
average	**0.604044**	0.541461	0.503667	0.550188	0.509337

(a)

Glass

	VM1	VM2	VM3	VM4	VM5
rare_matrix	0.477259	0.467146	0.363039	0.313882	0.465963
random_matrix	0.514916	0.513629	0.641634	0.650164	0.606015
average	0.496088	0.490388	0.502337	0.482023	**0.535989**

(b)

Page-blocks

	VM1	VM2	VM3	VM4	VM5
rare_matrix	0.720676	0.719496	0.671866	0.682976	0.685656
random_matrix	0.6567	0.65615	0.612202	0.619543	0.655472
average	**0.688688**	0.687823	0.642034	0.65126	0.670564

(c)

Vehicle

	VM1	VM2	VM3	VM4	VM5
rare_matrix	0.712325	0.621851	0.594291	0.644188	0.723903
random_matrix	0.541159	0.527543	0.699733	0.73221	0.707321
average	0.626742	0.574697	0.647012	0.688199	**0.715612**

(d)

Wine

	VM1	VM2	VM3	VM4	VM5
rare_matrix	0.812696	0.807971	0.81729	0.818441	0.813927
random_matrix	0.810475	0.809609	0.726201	0.819187	0.822311
average	0.811586	0.80879	0.771746	**0.818814**	0.818119

(e)

On special datasets with expert-defined benefit matrices, performance evaluation of voting methods is presented in Table 5.5. Here, it should be noted that in bank-loans, bankruptcy and lesion datasets, prediction of the minority class is much more important than prediction of majority class. In those domains, not surprisingly, probabilistic methods VM1 and VM2 produced the best results. This is because their internal structure relies solely on favoring prediction of minority classes. On the other hand, in arrhythmia2r dataset the important class which is the detection of arrhythmic instances is more abundant than healthy instances, and thus, VM1 and VM2 operate poorly in that dataset. On the average of all datasets, VM1 is the winner of benefit accuracy with a value of 0.74, but VM5 is very close with an average benefit accuracy of 0.72 on five of the datasets. When single voting is processed, VM5 is more preferable since it is more general and applicable to any kind of class distributions. However, if domain knowledge is utilized and it is seen that there is a great difference between class distributions of classes and that the minority class is more important, VM1 can be useful. Alternatively, if voting method is to be supported with benefit elements like generalization and pruning, then VM1 can be more effective, as we will see in Section 5.3.4.

Table 5.5: Behavior of single voting methods over special datasets

	VM1	VM2	VM3	VM4	VM5
arrhythmia2r	0.67106	0.638288	**0.814835**	**0.814835**	**0.814835**
bank-loans	**0.680171**	0.66353	0.669783	0.512857	0.551286
bankruptcy	0.503108	0.503108	0.510482	0.509069	**0.510671**
dermatology	**0.969257**	**0.969257**	0.700129	0.58594	0.936078
lesion	**0.878923**	**0.878923**	0.673071	0.724255	0.797718
Average	**0.740504**	0.730621	0.67366	0.629391	0.722118

5.3.2 Effect of Generalization

Merging formed intervals is important for reducing the problem size and the effect of overfitting to the training data. It also helps the algorithm become more general and effective over a wide range of values. In this subsection, we present how the generalization strategies mentioned in Section 4.2.3 change the total benefit and benefit accuracy with respect to those of single voting schemes.

Effect of Merging Intervals that Have the Same Frequent Class (SF)

Merging intervals that have the same frequent class in the interval is a simple way of merging intervals, yet this strategy does not consider benefit calculations of the interval. It simply joins two intervals into one by looking at the most frequent class in these intervals. Table 5.6 lists the changes of total benefit in terms of ratio (i.e., change in benefit/prior benefit) when this form of generalization takes place. For example, a value of 3.29 means that the increase in total benefit is 3.29 times the prior total benefit. In the following tables, (a)'s near multi-class datasets represent the results obtained when using rare class benefit matrices.

Table 5.6: Changes in total benefit when SF is used on single voting methods

Datasets	Voting methods				
	VM1	VM2	VM3	VM4	VM5
breast-cancer-w	-0.042	-0.070	-0.113	-0.030	-0.001
diabetes	0.202	-0.079	-0.018	0.092	-0.012
ionosphere	8.330	8.045	-0.001	0.042	0.012
liver	0.007	-1.716	0.000	0.011	0.008
sonar	114.857	104.857	1.038	1.030	1.850
Ecoli (a)	-0.939	-4.900	-3.500	-0.114	0.414
glass(a)	0.721	0.317	1.145	0.322	0.608
page-blocks(a)	0.187	0.078	0.104	-0.010	0.085
vehicle(a)	0.000	-1.261	2.439	0.272	-0.083
wine(a)	0.239	0.044	0.199	0.142	0.241
arrhythmia2r	0.299	1.155	0.000	0.000	0.000
bank-loans	-0.348	-1.432	0.145	-0.038	0.127
bankruptcy	-1.138	-1.257	-0.449	-0.732	-0.646
dermatology	-0.027	-0.452	-0.001	0.000	-0.033
lesion	0.030	-0.458	0.000	0.020	0.063
# datasets in which benefit is increased	9	6	6	8	9

By looking at the resultant values in Table 5.6, it can be said that the effect of merging intervals that have the same frequent class is not very promising when used with VM2 and VM3. However, on 9 of 15 datasets SF technique has produced an increase when used with VM1 and VM5. Especially on sonar dataset, which has a more or less

equal distribution of classes, joining intervals of same frequent class with VM1 and VM2 produced highly effective results. As our empirical results on datasets show, SF option is more useful when used in combination with SBC and HC generalization methods.

Effect of Merging Intervals That Have the Same Beneficial Class (SBC)

According to the results presented in Table 5.7, it is often beneficial to use SBC merging strategy to generalize intervals, especially when using VM1,VM2 and VM4 voting. It can be seen that in VM4 column, the number of benefit increase situations is more than any other voting methods. This suggests that, if VM4 is used, it will be beneficial to use SBC merging most of the time. However, when VM3 and VM5 is in use, it is questionable whether to use SBC or not, since in only 6 out of 15 datasets, it has caused an increase in the benefit obtained.

Table 5.7: Effect of SBC on single voting methods

	VM1	VM2	VM3	VM4	VM5
breast-cancer-w	-0.028	-0.028	-0.059	-0.029	-0.001
diabetes	-0.003	1.124	-0.019	0.038	-0.026
ionosphere	8.723	8.714	-0.013	0.032	0.000
liver	0.526	1.280	0.000	0.011	-0.012
sonar	92.286	100.286	0.953	1.026	1.448
Ecoli (a)	-0.888	-1.750	-0.097	0.684	0.155
glass(a)	0.868	0.994	0.984	0.372	0.741
page-blocks(a)	0.092	0.085	0.074	-0.062	-0.003
vehicle(a)	0.035	0.049	2.742	0.227	-0.155
wine(a)	0.268	0.081	0.238	0.268	0.264
arrhythmia2r	0.145	-1.733	0.000	0.000	0.000
bank-loans	-0.262	-0.751	-0.087	0.002	0.208
bankruptcy	-0.873	-0.874	-0.865	-0.864	-0.865
dermatology	-0.009	-0.166	0.038	0.000	-0.043
lesion	0.016	0.017	0.000	0.020	0.069
# datasets in which benefit is increased	**9**	**9**	**6**	**10**	**6**

Effect of Merging Intervals That Have High Confidence (HC)

Merging intervals that have high confidence is a probabilistic method that does not consider benefit characteristics of classifications. Table 5.8 presents HC's effect on single voting methods.

Table 5.8: Effect of HC on single voting methods

	VM1	VM2	VM3	VM4	VM5
breast-cancer-w	-0.007	-0.132	-0.091	-0.012	-0.001
diabetes	0.262	0.559	-0.022	0.070	0.014
ionosphere	0.089	0.089	-0.002	0.000	0.008
liver	-0.713	-0.159	0.000	0.011	-0.003
sonar	-16.286	12.000	1.111	1.148	1.047
Ecoli (a)	-0.304	1.750	-3.097	0.494	0.241
glass(a)	0.253	0.584	-2.097	0.004	0.766
page-blocks(a)	0.036	0.073	-0.050	0.002	0.048
vehicle(a)	0.023	1.005	2.742	0.578	-0.174
wine(a)	0.112	0.183	0.124	0.202	0.157
arrhythmia2r	-0.212	0.857	0.000	0.000	0.000
bank-loans	0.003	-0.494	-0.327	-0.081	-0.048
bankruptcy	0.114	0.817	1.939	1.755	1.888
dermatology	0.000	-0.500	-0.240	-0.825	-0.159
lesion	0.015	0.028	0.000	0.020	0.067
# datasets in which benefit is increased	**9**	**11**	**4**	**10**	**9**

Overall, HC operation is effective with all of the voting methods except VM3. Especially, it enhances the accuracy of VM2, i.e., causing an increase of benefit in the eleven datasets. This improvement power of HC can be boosted up by using it in combination with other parameters.

Effect of Merging Intervals That Have High Benefit Confidence (HBC)

The most profitable of all generalization strategies is HBC as the experimental results presented on Table 5.9 demonstrate. These results indicate that HBC is a powerful generalization process effective over all voting methods in general. Especially combined

with VM4 and VM5, HBC increases total benefit in most of the datasets. Results emphasize that an effective BMFI version should use HBC as an internal routine.

Table 5.9: Effect of HBC on single voting methods

Datasets	Voting methods				
	VM1	VM2	VM3	VM4	VM5
breast-cancer-w	0.001	-0.024	-0.013	-0.001	0.000
diabetes	0.868	-0.678	0.002	0.091	0.030
ionosphere	-0.607	-0.955	0.001	0.031	0.023
liver	-0.394	0.602	0.011	0.022	0.008
sonar	122.286	119.714	1.287	1.372	2.176
Ecoli (a)	-0.708	-0.650	-0.855	-0.051	0.121
glass(a)	0.489	0.646	-0.855	0.042	0.361
page-blocks(a)	0.093	0.085	0.041	0.048	0.003
vehicle(a)	0.165	1.488	2.470	0.454	-0.039
wine(a)	0.199	0.185	0.114	0.144	0.145
arrhythmia2r	-0.415	-2.683	0.000	0.000	0.000
bank-loans	-0.003	-0.316	0.003	0.033	-0.042
bankrupcty	2.640	2.605	2.593	2.604	2.572
dermatology	0.000	0.000	0.011	0.000	0.010
lesion	0.010	0.007	0.000	0.020	0.047
# datasets in which benefit is increased	9	8	10	11	11

5.3.3 Effect of Pruning

Pruning, as described in 4.2.4, is helpful when there is an excessive amount of overfitting in classification model besides the noisy and redundant data in the domain. Our experiments with BMFI algorithm also confirm this fact. However, it should be noted that pruning is not very effective without generalization. Table 5.10 gives an idea about the change in benefit accuracy when pruning is used solely over the voted intervals.

As it can be seen in Table 5.10, pruning without generalization of intervals does not change benefit accuracy especially when used with beneficial voting methods. Moreover, it has a degrading effect over VM5. This is due to the internal structure of the pruning methodology. Pruning is based on divergence in majority and beneficial classes of

intervals and in intervals formed with beneficial voting methods, these two are generally equal. That's why pruning is more successful when used with probabilistic voting schemes.

Table 5.10: Effect of pruning on voting methods

Datasets	Voting methods				
	VM1	VM2	VM3	VM4	VM5
breast-cancer-w	0.009	0.009	0.000	-0.001	0.009
diabetes	1.252	2.629	0.000	0.005	0.001
ionosphere	6.875	6.875	0.000	0.022	0.000
liver	3.979	4.152	0.000	0.000	0.000
sonar	13.143	13.143	0.000	-0.005	0.000
Ecoli (a)	-0.128	-0.400	3.484	1.861	-0.121
glass(a)	0.000	0.143	0.177	0.724	0.000
page-blocks(a)	-0.005	-0.002	0.127	0.071	0.000
vehicle(a)	0.035	2.044	4.500	0.089	0.000
wine(a)	0.000	0.001	0.000	0.000	0.000
arrhythmia2r	2.161	4.957	0.000	0.000	0.000
bank-loans	0.060	0.131	0.000	0.381	0.029
bankrupcty	-0.004	-0.004	0.000	0.000	0.000
dermatology	-0.005	-0.020	-0.172	0.000	-0.005
lesion	0.100	0.028	0.154	-0.127	0.058
# datasets in which benefit is increased	9	11	5	7	4

Alternatively, when pruning is employed after merging intervals with respect to their beneficial classes and benefit confidences, i.e., with SBC and HBC, more profitable results are obtained. Results of these joint effects are given in Table 5.11. Although the increase in benefit accuracy is relatively lower in the case of VM5, there happens to be a promising overall increase when pruning is used with probabilistic voting. With VM3, in eleven of fifteen datasets there is a slight increase in total benefits, whereas with VM2 this increase is more significant in thirteen of the datasets. The results reinforce the hypothesis that with most of the datasets, it is beneficial to use a combination of SBC, HBC and pruning options.

Table 5.11: Joined effect of SBC, HBC and pruning on voting methods

	VM1	VM2	VM3	VM4	VM5
breast-cancer-w	-0.028	-0.028	-0.059	-0.029	-0.001
diabetes	2.085	3.842	-0.019	0.040	0.006
ionosphere	8.723	8.723	-0.029	-0.017	-0.047
liver	5.377	5.377	0.011	0.022	0.019
sonar	45.714	56.857	1.287	1.372	2.176
Ecoli (a)	-0.708	3.550	3.855	3.671	-1.190
glass(a)	1.168	1.292	1.371	1.682	0.734
page-blocks(a)	0.232	0.231	0.175	0.103	-0.644
vehicle(a)	0.174	2.719	6.152	0.827	-0.377
wine(a)	0.333	0.320	0.266	0.223	0.198
arrhythmia2r	2.132	4.733	0.000	0.000	0.000
bank-loans	0.479	0.290	0.566	0.327	-0.120
bankruptcy	2.664	2.664	2.593	2.604	2.572
dermatology	-0.001	-0.005	0.041	0.000	0.005
lesion	0.120	0.086	0.034	-0.127	0.084
# datasets in which benefit is increased	**12**	**13**	**11**	**10**	**8**

5.3.4 Overall Evaluation

Up to now, we have observed that if no generalization or pruning is used, then single beneficial voting methods that incorporate benefit knowledge directly are effective over any kind of datasets. On the other hand, probabilistic voting methods can be quite effective when supported with benefit-based generalization and pruning. In this section, we present overall outcomes of strategies tested on different datasets and discuss about which form of BMFI is more profitable in general. For this reason, the best four of progression series of results for each single dataset have been plotted and displayed in Figure 5.2, Figure 5.3 and Figure 5.4. In these figures, ‘single’ denotes the voting method used singly and ‘p’ option represents the employment of pruning,

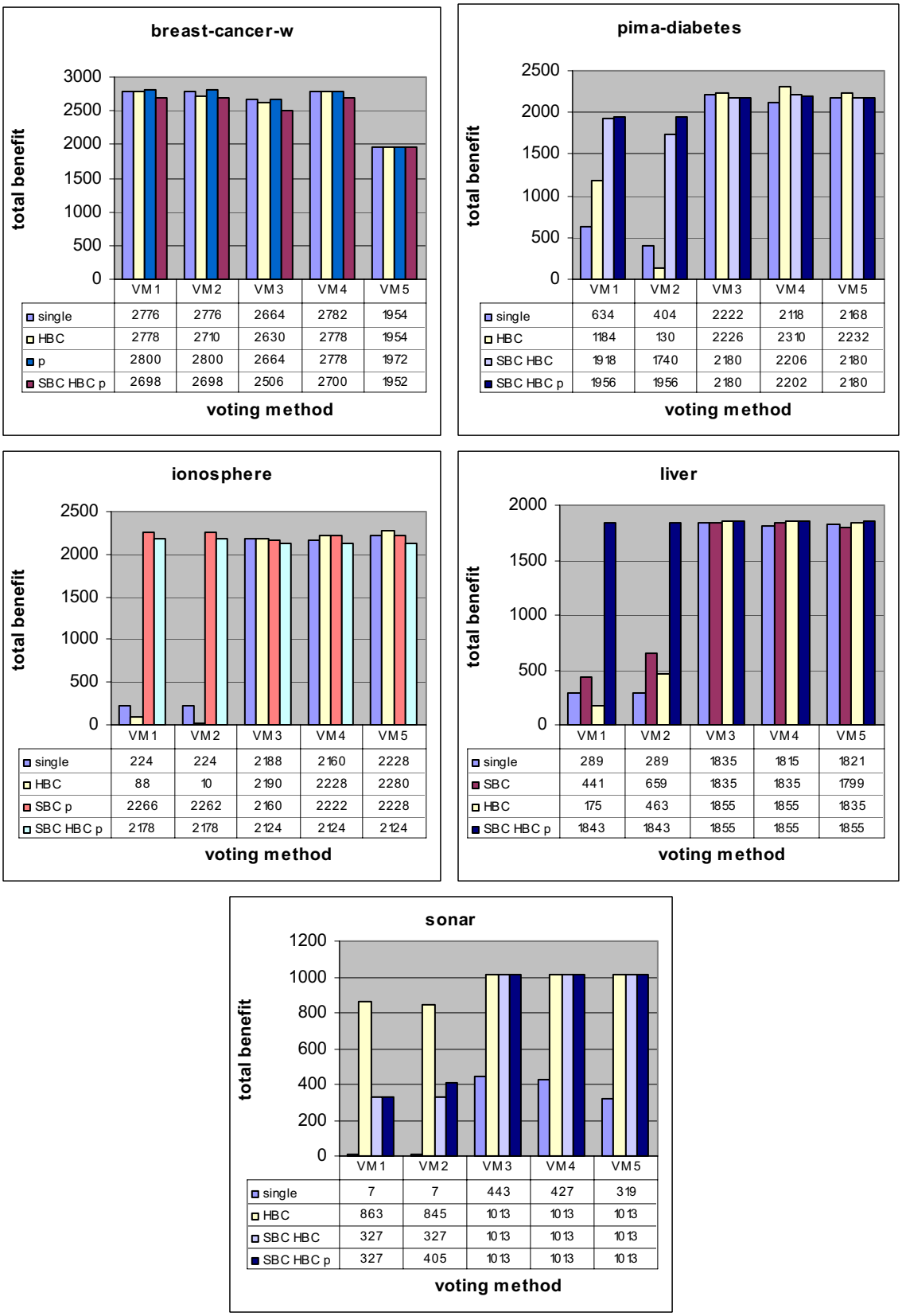


Figure 5.2: Overall BMFI progressions on two-class benchmark datasets

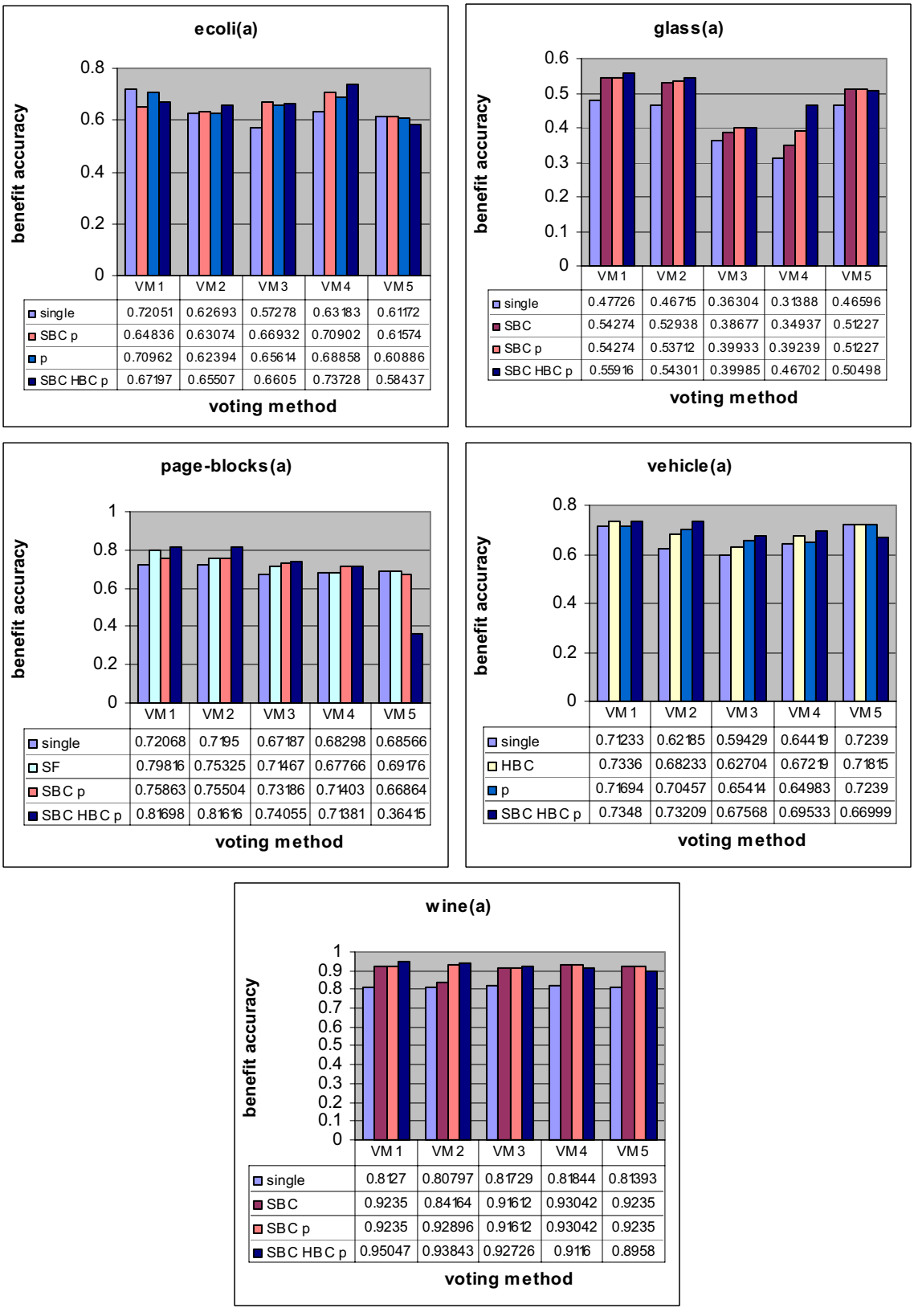


Figure 5.3: Overall BMFI progressions on multi-class benchmark datasets

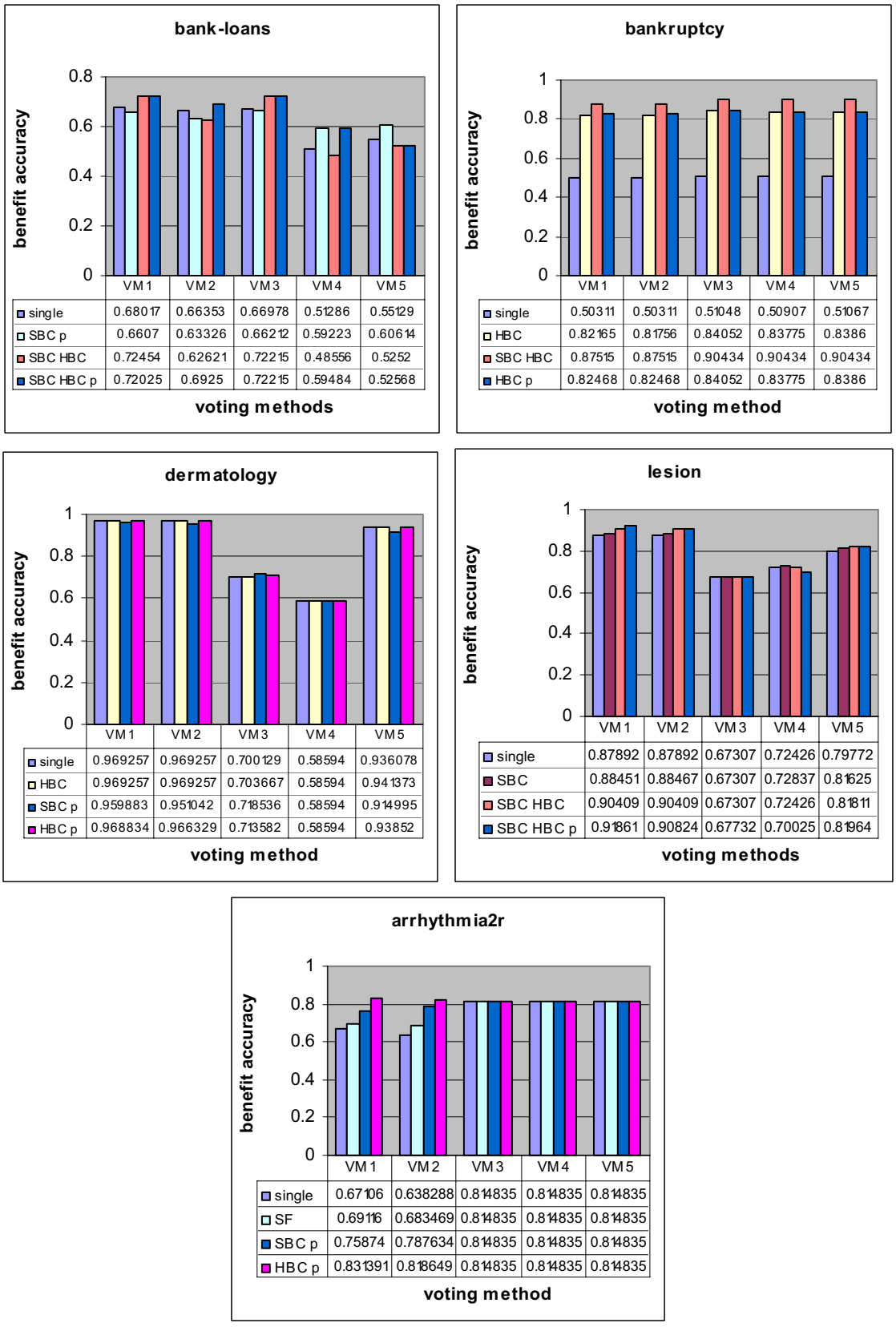


Figure 5.4: Overall BMFI progressions on special datasets

For two-class datasets, except breast cancer domain, beneficial voting methods dominate probabilistic voting and VM4 and VM5 can be counted as the most efficient voting methods in maximizing the benefit. VM3 also displays very close results to VM4 and VM5.

In Figure 5.3, multi-class datasets are evaluated with benefit matrices that favor the prediction of the rare class. The results indicate that probabilistic methods VM1 and VM2 supported with benefit-based generalization and pruning, are efficient in predicting minority classes and thus in acquiring higher benefit accuracy. On the other hand, if benefit matrices that assign benefits independent of the distribution of the classes are used, VM4 and VM5 become more effective in increasing benefit accuracy. In all of special datasets except arrhythmia2r, predefined benefit matrices also imply that minority class is more important. Results over those datasets represented in Figure 5.4 again denote the dominance of VM1 and VM2 in benefit performance. Respectively, VM3 is not as effective as the other voting methods, however it achieves promising results especially on bank-loans domain.

To sum up, the performance of the voting methods relies mostly on the nature of the benefit matrix. If the correct prediction of infrequent classes is more profitable, then probabilistic voting with benefit-based generalization and pruning is more preferable. In this point, it is remarkable that using just the voting method for classification is not sufficient in the case of probabilistic voting. When used, they should be supplemented with interval merge and prune operations. As the empirical results we achieved on our datasets imply, and after examining other possible combinations of options, we propose to use VM1 voting together with SF, BC and HBC techniques when benefit of classification of a certain class is in inverse proportion to the distribution of that class. In other words, if the rarer the class label, the more profitable it is, then, best results are obtained using VM1 with SF, SBC and HBC. Of course, further experimentation on diverse datasets, will outline the borders of this generalization more precisely.

On the contrary, using benefit information directly in voting method, i.e., beneficial voting is a more general approach and applicable to any sort of benefit matrix. VM4 and

VM5 display very close performance results in this framework. When the benefit matrix is not dependent on the distribution of classes in the dataset, either of VM4 or VM5 can be employed together with SBC, HBC and pruning to boost up the benefit performance.

5.4 BMFI versus Other Cost-Sensitive Algorithms

In this section, we compare BMFI results with wrapper cost-sensitive strategies. For observing relative performance of our algorithmic approach, we have compared BMFI with MetaCost and CostSensitiveClassifier of Weka [6] on base classifiers Naïve Bayesian Classifier and C4.5 (J4.8) decision tree learner. In this section, after giving a brief outline of the specifications of those algorithms, their benefit performances on several datasets are presented. All results given are recorded by using 10-fold cross validation over the whole datasets.

5.4.1 Properties of Comparison Algorithms

In Table 5.12, list of algorithms that we have used for comparison purposes is presented. In the rest of this thesis, we will use the pseudonyms for easy-referencing.

Table 5.12: List of cost-sensitive algorithms used for evaluation

Pseudonym	Description
MetaNB	MetaCost on Naive Bayes
MetaJ48	MetaCost on J4.8
C1NB	CostSensitiveClassifier with reweighting on Naive Bayes
C2NB	CostSensitiveClassifier with direct minimization on Naive Bayes
C1J48	CostSensitiveClassifier with reweighting on J4.8
C2J48	CostSensitiveClassifier direct minimization on J4.8
C1VFI	CostSensitiveClassifier with reweighting on VFI
C2VFI	CostSensitiveClassifier with direct minimization on VFI

5.4.1.1 MetaCost

MetaCost is a wrapper algorithm that takes a base classifier and makes it sensitive to costs of classification [16]. It operates with a bagging logic beneath and learns multiple classifiers on multiple bootstrap replicates of the training set. By using the resultant votes of classifiers, MetaCost relabels training instances with the estimated optimal class. Pseudo-code and other algorithmic details of MetaCost have been given in section 3.1.3 of this thesis.

5.4.1.2 Weka.CostSensitiveClassifier

There are two methods implemented in Weka's cost-sensitive wrapper algorithm. First method uses reweighting of training instances in order to make its internal classifier cost-sensitive. The second method requires its internal classifier to be a distribution based classifier and makes direct cost-minimization based on probability distributions. This is very similar to Zadronzy et al.'s studies presented in [51].

For the first method of Weka's cost-sensitive classifier, we use pseudonym C1 and for the second C2, respectively. In C1, if the internal classifier has not the property of weight instances, then resampling is done to adjust weights of the training instances. On the other hand, if internal classifier supports weighting of instances, then the weights of the instances are simply updated to reflect the effect of benefit of classification.

When C2 is wrapped around a distribution based classifier, it uses the probability outputs of the internal classifier and puts these probabilities in the optimal decision equation to determine the optimal predictions that minimizes the cost of classification.

5.4.1.3 Naive Bayesian Classifier (NBC)

We have chosen naive Bayesian classifier as the first internal classifier to be used with MetaCost, C1 and C2 due to its simplicity and accuracy. Naive Bayesian classifier is based on Bayes theorem and on the simplifying assumption that the feature values are conditionally independent given the target value [38]. This assumption is the same as the one that feature intervals concept relies on. Although this may not be a realistic

assumption, it produces highly practical and efficient results and in some domains, Naive Bayesian Classifier's performance has been shown to outperform to that of neural network and decision tree learning.

5.4.1.4 J4.8 Decision Tree Learner

J4.8 is the Weka's implementation of C4.5 decision tree learner. It actually implements a later and a slightly improved version called C4.5 Revision 8, which was the last public version of this family of algorithms before C5.0 [50]. C4.5 is proven to be a very successful decision tree learner [41]. That's why most of the cost-sensitive research presented in Section 3.2.1 has focused on increasing benefit performance of this classification algorithm.

5.4.1.5 Voting Feature Intervals (VFI) Classifier

VFI is another feature projection based classifier [31] and its underlying concept representation (i.e., feature intervals) is the same as BMFI. In Weka's implementation of VFI, there is a simple feature weighting scheme added. Higher weight is assigned to more confident intervals where confidence is a function of entropy such that

$$\text{Weight}(f_i) = (\text{entropy of class distribution on } f_i \,/\, \text{maximum uncertainty})^{bias}$$

where *bias* represents the strength of bias towards more confident features. In our experiments, we have used the default bias value of 0.6 with feature reweighting.

5.4.2 Comparative Results

Results obtained by running eight algorithms discussed on binominal, multi-class and special datasets are presented in Table 5.13(a)-(e) and Table 5.14. In Table 5.13, the behavior of cost-sensitive algorithms with respect to benefit ratios and their corresponding total benefit values are presented. Average values in the bottom of each sub-table is the average total benefit value each algorithm acquire in the end of using five pre-defined benefit ratios (represented with column named *b* in the tables).

Table 5.13: Total benefit values for different benefit ratios on two-class datasets

Breast-Cancer Wisconsin Dataset

b	MetaNB	MetaJ48	C1NB	C2NB	C1J48	C2J48	C1VFI	C2VFI	BMFI
2	890	862	900	900	852	836	860	24	896
5	1599	1551	1601	1611	1555	1471	1511	747	1603
10	2784	2678	2776	2796	2692	2580	2596	1952	2722
20	5154	5016	5122	5206	4928	4846	4766	4362	5172
50	12262	12118	12264	12374	12240	11794	11276	11592	12324
AVG	4537.8	4445	4532.6	4577.4	4453.4	4305.4	4201.8	3735.4	4551.4

(a)

Pima-diabetes Dataset

b	MetaNB	MetaJ48	C1NB	C2NB	C1J48	C2J48	C1VFI	C2VFI	BMFI
2	398	494	508	460	474	466	36	36	370
5	958	940	1128	992	946	920	-394	834	958
10	2136	2180	2316	2098	2240	1932	-1114	2164	2100
20	4466	4486	4736	4690	4706	4328	-2554	4824	4866
50	12712	12404	12440	12766	12512	11324	-6874	12804	12902
AVG	4134	4100.8	4225.6	4201.2	4175.6	3794	-2180	4132.4	4239.2

(b)

Ionosphere Dataset

b	MetaNB	MetaJ48	C1NB	C2NB	C1J48	C2J48	C1VFI	C2VFI	BMFI
2	472	504	482	488	470	458	530	378	476
5	1043	1159	1103	1083	1117	1019	1175	999	1165
10	1986	2284	2142	2088	2294	1994	2250	2124	2280
20	3876	4450	4414	4078	4506	3904	4400	4374	4516
50	9544	11172	11150	10146	11124	9634	10850	11124	11256
AVG	3384.2	3913.8	3858.2	3576.6	3902.2	3402	3841	3799.8	3938.6

(c)

Liver Disorders Dataset

b	MetaNB	MetaJ48	C1NB	C2NB	C1J48	C2J48	C1VFI	C2VFI	BMFI
2	251	215	239	267	233	245	227	253	253
5	841	775	827	865	831	665	701	847	855
10	1819	1819	1785	1865	1855	1515	1491	1837	1857
20	3819	3759	3745	3865	3855	3259	3071	3817	3855
50	9759	9579	9465	9865	9855	8315	7811	9757	9855
AVG	3297.8	3229.4	3212.2	3345.4	3325.8	2799.8	2660.2	3302.2	3335

(d)

Sonar Dataset

b	MetaNB	MetaJ48	C1NB	C2NB	C1J48	C2J48	C1VFI	C2VFI	BMFI
2	107	185	207	155	131	161	26	109	127
5	296	444	484	336	468	354	27	376	458
10	687	955	929	827	963	691	1	821	1013
20	1695	1975	1965	1803	1995	1357	-29	1711	2123
50	4547	5309	5053	4969	5453	3649	-119	4381	5453
AVG	1466.4	1773.6	1727.6	1618	1802	1242.4	-18.8	1479.6	1834.8

(e)

Results presented in Table 5.13 are average BMFI runs with voting method VM1, generalization strategies SBC and HC together with pruning. If we look at the average values, it is observed that BMFI is very effective in maximizing the benefit in all of the two-class datasets. At this point, it is worth stating that as benefit ratio increases, i.e., classification of a certain class becomes more important, the success of BMFI increases. This is an important highlight of the BMFI algorithm and is mostly due to BMFI's high sensitivity to benefits of classification.

In breast-cancer, ionosphere, and sonar datasets BMFI shows the best benefit performance on the average. In breast-cancer BMFI is the second best and in liver disorders dataset, it is the third. In breast-cancer and liver disorders datasets C2NB performs better than BMFI. This ranking can be counted as very successful from BMFI's point of view, because it outperforms most of the algorithms, among which there is a successful decision tree learner C4.5 that considers conditional cases between features themselves. Here, it should not be forgotten that for different benefit ratios, there exists different performance winners and this observation emphasizes that benefit maximization in a particular domain is mostly dependent of the nature of the benefit matrix.

In addition, it is worthwhile to note that BMFI outperforms cost-sensitive versions of VFI (C1VFI and C2VFI), which is also a feature-projection based classifier. This observation suggests that using benefit knowledge inside the algorithm itself is more effective than wrapping a meta-stage around it to transform it into a cost-sensitive classifier.

In Table 5.14, results obtained by BMFI are compared to other algorithms. The results have been compiled by making use of the rule-of-thumb presented in the end of

Section 5.3. According to this rule, when the infrequent class is important in the dataset, BMFI is executed using VM1 with SF, SBC and HBC. If importance is not related to distribution of classes, then VM4 and VM5 is tried and best result is recorded. Results over binominal datasets are the ones achieved when using a benefit ratio of 10. For multi-class datasets, the average results over two benefit matrices, i.e., random and rare class, are presented. In sonar, bankruptcy and lesion domains, BMFI is the winner of performance whereas in bank-loans and dermatology datasets its performance is very high and comparable to other algorithms. However, in ecoli database, BMFI performs poorly, mostly due to the nature of the specified domain. In glass dataset, BMFI was the fifth in performance. In vehicle and wine domains, BMFI gains fourth place in the benefit performance ranking. It can be observed that no algorithm is dominant over all domains and performance is highly dependent on the nature of the domain. Yet, if we look at overall results in Table 5.14, C2NB, C1J48 and BMFI are the most efficient approaches in terms of benefit, and they have both achieved best performance on three of the datasets. Another interesting observation that can be derivable from Table 5.14 is that, no single algorithm has shown concrete benefit performance on multi-class domains and for each of the six multi-class datasets, a different algorithm has been the most successful in maximizing the total benefit.

Table 5.14: Comparative evaluation of BMFI with wrapper cost-sensitive algorithms

	MetaNB	MetaJ48	C1NB	C2NB	C1J48	C2J48	C1VFI	C2VFI	BMFI
breast-cancer-w	2784	2678	2776	2796	2692	2580	2596	1952	2722
pima-diabetes	2136	2180	2316	2098	2240	1932	-1114	2164	2100
ionosphere	1986	2284	2142	2088	2294	1994	2250	2124	2280
liver disorders	1819	1819	1785	1865	1855	1515	1491	1837	1857
sonar	687	955	929	827	963	691	1	821	1013
bank-loans	-1080	-608	-1288	-816	104	-744	-1656	-4016	-180
bankruptcy	11225	10789	11088	10624	10865	10595	11141	11271	11399
dermatology	2747	2631	2747	2757	2623	2658	2532	2049	2701
lesion	2479	2223	2524	2551	2219	2179	1827	1139	2560
ecoli(avg)	1098	1177.5	1069.5	1101	1165	1110.5	1261.5	1282	976.5
glass(avg)	607.75	814.5	539.75	618.25	766.25	713.25	707.75	421	643.75
vehicle(avg)	2073	3597	2392.5	2378.5	3681	3156.5	1089.5	2165.5	3115
wine(avg)	1985	1789.5	1966	1981.5	1863	1719	1937.5	1672.5	1960

5.5 Feature-Dependent Classification using BMFI

As described in Section 3.3 and Section 4.2.2, benefit of a classification can be quite dependent on one or more of the domain features. In such a case, there is no static benefit matrix available and the classification of each instance has a different benefit associated with it. In this thesis, we have presented a naïve way for dealing with such situations and indicate that this is an open area for extensive research. Here, it should be noticed that it is not straightforward to incorporate feature-dependency concept to a framework like feature projections which assume independency in feature domain.

Bank-loans data is a direct application area for feature dependency. If the benefit matrix is assigned so as to indicate the net cash flow in the bank with respect to granted loans, then for each customer asking for a loan, there is a different benefit dependent on the amount requested. This situation can be formulated by the benefit matrix as follows:

	Actual class	
Prediction	default	Don't default
default	$r \times la$	$-la$
Don't default	$-r \times la$	0

Here, r is the interest rate that the bank utilizes, logically $0<r<1$, and la is the loan amount that the customer asks for. According to this matrix, if the money is granted and customer pays the loan back, then net money gain from bank's perspective is $r \times la$. If money is not granted to a good customer who will pay it back, this means the bank has lost $r \times la$ amount of profit. On the contrary, if the loan is granted but the customer does not pay it back, then the bank loses the entire loan amount. The net cash flow is 0 when there is no money is given to a bad customer who will not default.

Figure 5.5 illustrates the benefit accuracy of BMFI and its change with respect to interest rate that the bank uses for loan applications. In this chart, the results indicate that the lower the interest rate, the higher benefit accuracy BMFI acquires. Since most of the Turkish banks applies a combined interest policy with an interest rate around 0.08 per

month, then we can say that overall accuracy of BMFI on bank-loans domain is approximately 0.78.

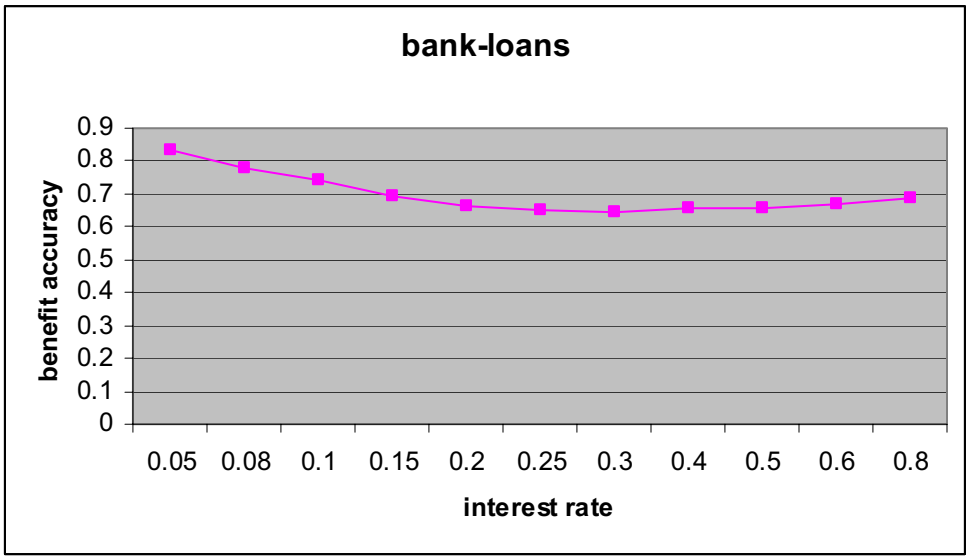


Figure 5.5: Change in benefit accuracy with respect to interest rate in bank-loans domain

As the interest rate increases in this experiment, the ratio of importance between classes decreases and this causes a decrease in the performance of BMFI, since it is already observed that BMFI performs better when the benefit ratio between classes becomes higher.

Chapter 6

Conclusion and Future Work

In this thesis, we have focused on the problem of making predictions when the possible outcomes have different benefits associated with them. We have implemented a new methodology, namely BMFI that uses the predictive power of feature intervals concept in maximizing the total benefit gained in a classification problem. The previous studies in the literature of cost-sensitive learning have undervalued the differences in the benefits of correct classifications. For this reason, as the main contribution of the thesis, we aimed to emphasize the importance of classification benefits and to present the first study, as to our best knowledge, in the literature that is built solely on benefit concept and its maximization.

Classification in feature projection based classifiers depends on a voting process within the formed feature intervals. In the framework of BMFI, we have proposed five different voting methods that are shown to be effective over different domains. In addition, a number generalization and pruning methodologies based on benefits of classification are implemented and experimented. The results obtained gave us an insight of using different techniques, dependent on the characteristics of the domain.

Since BMFI is a non-incremental inductive concept learning algorithm, some information about the domain, e.g., class distributions and type of features, is known apriori. By making use of this knowledge, algorithmic parameters of BMFI can be

arranged in such a way that enables fine-tuning with respect to specifications of the domain. We have observed that the benefit maximization routine is highly dependent on the benefit matrix introduced and the corresponding class distributions in the domain. Therefore, we propose using VM1 voting method with SF, SBC and HBC techniques when the minority class prediction is more beneficial. On the other hand, our results demonstrated that when benefits are not related to class frequencies, then it would be more apt to employ VM4 or VM5 voting methods.

When a probabilistic voting method is used solely in BMFI, then it becomes an error-based classifier. The results show that BMFI is very effective in maximizing the total benefit compared to its error-based version.

Furthermore, BMFI has been compared to MetaCost and other two cost-sensitive classification algorithms implemented in the Weka package. These two generic algorithms are wrapped over two prevailing classification algorithms, NBC and C4.5. BMFI results are very promising when compared to MetaCost, C1 (instance reweighting) and C2 (direct cost minimization techniques). Individual characteristics of the datasets influence results significantly, due to the extreme correlation between cost-sensitivity and class distributions. It can be inferred from the results that no algorithm is superior to the others in all of the domains. This observation suggests that there is still need for future research and improvement in cost-sensitive classification field.

Another contribution of this study is the proposal of a new metric, namely benefit accuracy, for the cost-sensitive evaluation of classifiers. It computes the relative accuracy of the total benefit obtained with respect to the maximum possible benefit achievable in the domain. Benefit accuracy metric is the generalization of the classical predictive accuracy metric. It is easy to interpret since it resembles the standard predictive accuracy.

In the context of this study, we have also dealt with situations when benefits are not static and dependent on the values of features. We have presented a naive approach concerning this issue and experimented over a recently constructed dataset, bank-loans data. We have achieved promising results in this domain as well.

The research described in this thesis can be extended in many directions. First of all, by testing over new domains, BMFI options can be stabilized more and by this means, the results can be improved. Stratified cross-validation, in which the folds are stratified so that they contain approximately the same proportions of class labels as the original dataset, can be employed. We think that this will greatly enhance the accuracy of the algorithm.

As an additional future work, feature-dependent domains can be explored in depth and feature-dependency aspect of BMFI can be improved. Especially, new voting methods that are more efficient in handling functions of varying benefits can be developed. In addition, benefit maximization can be extended to include the feature costs. In order to accomplish this, feature selection mechanisms that are sensitive to individual costs of features can be utilized. This will make the classification algorithm more comprehensive and applicable in real-world domains. Furthermore, this sort of benefit maximization research can be extended to handle incremental datasets, as in the case of active learning.

Bibliography

[1] PredictionWorks Data Mining Glossary, http://www.predictionworks.com/glossary.

[2] Boosting Research Site, http://www.boosting.org.

[3] Cost-sensitive learning bibliography. Online bibliography. P. Turney, O. Boz, editors, Institute for Information Technology of the National Research Council of Canada, Ottawa, 1997, http://home.ptd.net/~olcay/cost-sensitive.html.

[4] ICML-2000 Workshop on Cost-Sensitive Learning – Workshop Notes, http://www.dmargineantu.net/Workshops/Workshop-ICML2000/worknotes.html.

[5] KDD Cup 1998, http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html.

[6] The University of Waikato software documentation, Weka 3 - Data Mining with Open Source Machine Learning Software in Java. [http://www.cs.waikato.ac.nz/~ml/weka].

[7] C. L. Blake and C. J. Merz. UCI repository of machine learning databases. University of California, Irvine, Department of Information and Computer Sciences, 1998. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

[8] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C.E. Brodley. Pruning decision trees with misclassification costs. In *Proceedings of Tenth European Conference on Machine Learning (ECML-98)*, pages 131-136, Berlin, 1998.

[9] R.A. Brealey and S.C. Myers. *Principles of Corporate Finance*. New York, NY: MacGraw-Hill, 5th edition, 1996.

[10] L. Breiman. Bagging Predictors. *Machine Learning,* 24(2):123-140, 1996.

[11] L. Breiman, J. H. Friedman, R. A. Olsen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, Wadsworth, 1984.

[12] P. Chan and S. Stolfo. Towards scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164-168, 1998.

[13] P. Chan and S. Stolfo. Learning with Non-uniform Class and Cost Distributions: Effects and a Distributed Multi-Classifier Approach. In *Workshop Notes KDD-98 Workshop on Distributed Data Mining*, pages 1-9, 1998.

[14] G. Demiröz, and H. A. Güvenir. Classification by voting feature intervals. In *Proceedings of 9th European Conference on Machine Learning*, Maarten van Someren and Gerhard Widmer (Eds.), Springer-Verlag, LNAI 1224, pages 85-92, Prague, Czech Republic, April 23-25, 1997.

[15] P. Domingos. How to get a free lunch: A simple cost model for machine learning applications. In *Proceedings of AAAI-98/ICML-98 Workshop on the Methodology of Applying Machine Learning*, pages 1-7, Madison, Wisconsin, 1998.

[16] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 155-64, San Diego, CA, 1999.

[17] C. Drummond and R. Holte. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Proceedings of the 17th International Conference on Machine Learning (ICML'2000)*, pages 239-246, 2000.

[18] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[19] C. Elkan. Cost-sensitive learning and decision-making when costs are unknown. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000),* Stanford University, California, 2000.

[20] C. Elkan. The Foundations of Cost-Sensitive Learning. In *Proceedings of the Seventeenth International Joint Conference on Articial Intelligence*, August, 2001.

[21] N. Emeksiz. Diagnosis of gastric carcinoma tumors by multi-class voting. M.S. Thesis, Middle East Technical University, Turkey, July 2001.

[22] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: Misclassification cost-sensitive boosting. In *Machine Learning: Proceedings of the Sixteenth International Conference*, pages 97-105, San Francisco, Morgan Kaufmann, 1999.

[23] T. Fawcett and F. Provost. Adaptive fraud detection. *Journal of Data Mining and Knowledge Discovery*, 1(3):291-316, 1997.

[24] A. A. Freitas. On rule interestingness measures. *Knowledge-Based Systems*, 12(5-6): 309-315, October 1999.

[25] R. Herbrich. *Learning Kernel Classifiers*. The MIT Press, 2002. [http://www.learning-kernel-classifiers.org/index.htm].

[26] J. Hollmen, M. Skubacz, and M. Taniguchi. Input dependent misclassification costs for cost-sensitive classifiers. In *Proceedings of the Second International Conference on Data Mining*, pages 495-503, 2000.

[27] J. Gama. A cost-sensitive iterative Bayes. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000)*, Stanford University, California, 2000.

[28] H. A. Güvenir. Detection of abnormal ECG recordings using feature intervals. In *Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'2001)*, A. Acan, I. Aybay, and M. Salamah (Eds.), pages 265-274, Gazimagusa, T.R.N.C., 2001.

[29] H. A. Güvenir and H. G. Koç. Concept representation with overlapping feature intervals. *Cybernetics and Systems: An International Journal*, 29(3):263-282, 1998.

[30] H. A. Güvenir and I. Şirin. Classification by feature partitioning, *Machine Learning*, 23(1):47-67, 1996.

[31] H. A. Güvenir, G. Demiröz, and N. İlter. Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals. *Artificial Intelligence in Medicine*, 13(3):147-165, 1998.

[32] H. A. Güvenir, S. Altıngövde, I. Uysal, and E. Erel. Bankruptcy prediction using feature projection based classification. In *Proceedings of SCI/ISAS'99*, pages 108-113, Orlando, Florida, 1999.

[33] N. İkizler and H. A. Güvenir. Mining interesting rules in bank loans data. In *Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'2001)*, A. Acan, I. Aybay, and M. Salamah (Eds.), pages 238-246, Gazimagusa, T.R.N.C., June 2001.

[34] R. Kohavi and F. Provost. Glossary of terms. *Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Machine Learning*, 30:271-274, 1998.[http://robotics.stanford.edu/~ronnyk/glossary.html].

[35] F.Y. Lin and S. McClean. The prediction of financial distress using a cost sensitive approach and prior probabilities. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000),* Stanford University, California, 2000.

[36] D. Margineantu. On class probability estimates and cost-sensitive evaluation of classifiers. In *Workshop Notes, Workshop on Cost-Sensitive Learning, International Conference on Machine Learning*, June 2000.

[37] D. Margineantu. Methods for cost-sensitive learning. Ph.D. Dissertation, Oregon State University, 2002.

[38] T. Mitchell. *Machine Learning*, The McGraw-Hill Companies, Inc., 1997.

[39] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing misclassification costs: Knowledge-intensive approaches to learning from noisy data. In *Proceedings of the Eleventh International Conference on Machine Learning, ML-94,* pages 217-225, New Brunswick, New Jersey, 1994.

[40] F. J. Provost and A. P. Danyluk. A Study of Complications in Real-world Machine Learning. Available at http://citeseer.nj.nec.com/19634.html.

[41] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffmann, San Francisco, 1993.

[42] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions**.** *Machine Learning*, 37(3):297-336, 1999. Boston: Kluwer Academic Publishers.

[43] K. M. Ting. Cost-sensitive classification using decision trees, boosting and MetaCost. Book chapter in *Heuristic and Optimization for Knowledge Discovery*. Edited by Sarker, R., Abbass, H. & Newton, C. Idea Group Publishing, 2002.

[44] K. M. Ting. An instance weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659-665, May/June 2002.

[45] K. M. Ting and Z. Zheng. Boosting trees for cost-sensitive classifications. In *Machine Learning: ECML-98: 10th European Conf on Machine Learning*, pages 190-195, Chemnitz, Germany:Springer ,1998.

[46] P. Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000)*, pages 15-21, Stanford University, California, 2000.

[47] G. I. Webb. Cost-sensitive specialization. In *Proceedings of the 1996 Pacific Rim International Conference on Artificial Intelligence*, pages 23-34, Springer-Verlag.

[48] G. M. Weiss and F. Provost. The effect of class distribution on classifier learning. Technical Report ML-TR 43, Department of Computer Science, Rutgers University, 2001.

[49] W. Wilke and R. Bergmann. Considering decision cost during learning of feature weights. In *Proceedings of Advances in Case-Based Reasoning, Third European Workshop, EWCBR-96*, Lausanne, Switzerland. Lecture Notes in Computer Science, pages 460-472, 1168 Springer, 1996.

[50] I. H. Witten and E. Frank. Data Mining: Practical machine learning tools and techniques with Java implementations. Academic Press, 2000.

[51] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, AAAI Press (distributed by MIT Press), 2001.

[52] V.Bayer Zubek and T.G.Dietterich. Pruning improves heuristic search for cost-sensitive learning. To appear in *Proceedings of the International Conference on Machine Learning (ICML'02),* 2002.

Appendix A

UCI Benchmark Datasets

In the following, we provide the details for the benchmark datasets we have used from UCI Machine Learning Repository [7]. There is a total of ten benchmark datasets that have been used for evaluation in this thesis. Five of these datasets are two-class (binominal) datasets, and five of them are multi-class datasets. Along with dataset properties, benefit matrices that we have used in their experimentation are given.

A.1 Binary Datasets

Breast Cancer Wisconsin: This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. There are 9 features and 699 instances. The aim is to predict whether a taken tissue is benign or malignant. Missing attribute values in the entire dataset are 16. 458 (65.5%) of the instances belong to benign class and 241(34.5%) are malignant. From cost-sensitive point of view, detection of malignant instances are much more important.

Pima Indian Diabetes: The diagnostic, binary-valued variable investigated in this dataset is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). The population lives

near Phoenix, Arizona, USA. There are 768 instances and 8 features all of which are numeric valued. 500 of the instances are negative (tested negative for diabetes) and remaining 268 are positive.

Ionosphere: This is a radar data that was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.

Liver Disorders: The aim of this dataset is to predict whether a patient has liver disorders or not. There are 345 instances collected by Bupa Medical Research Ltd. The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the data file constitutes the record of a single male individual. The last feature represents the drink number of half-pint equivalents of alcoholic beverages drunk per day.

Sonar: This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occurs later in time, since these frequencies are transmitted later during the chirp. The label associated with each record in the original dataset contains the letter "R" if the object is a rock and "M" if it is a mine (metal cylinder). These class labels have been transformed to 0 and 1

for indicating the presence of a mine. 111 patterns recorded are mine and 97 patterns are rocks.

With two-class datasets, five different benefit matrices having benefit ratios of 2, 5, 10, 20 and 50 have been used in testing.

A.2 Multi-class Datasets

Ecoli: There are 336 instances with 7 predictive features in this dataset. The aim is to predict the protein localization sites within the cells. There are no missing attributes values and possible 8 localization sites.

Table A.1: Benefit table of ecoli dataset computed by using class probabilities in a way to favor minority class prediction

	Actual Class							
Prediction	C1	C2	C3	C4	C5	C6	C7	C8
C1	2	-2	-3	-4	-7	-29	-70	-70
C2	-1	3	-1	-2	-3	-15	-30	-30
C3	-1	-1	4	-2	-2	-10	-25	-25
C4	-1	-1	-1	5	-2	-7	-15	-15
C5	-1	-1	-1	-1	6	-4	-10	-10
C6	-2	-2	-2	-1	-1	9	-2	-2
C7	-3	-3	-2	-2	-1	-1	10	-1
C8	-4	-3	-2	-2	-1	-1	-1	10

Table A.2: Random benefit table of ecoli dataset with a ratio of 2 between consecutive class labels.

	Actual Class							
Prediction	C1	C2	C3	C4	C5	C6	C7	C8
C1	1	-2	-2	-3	-3	-3	-3	-3
C2	-1	2	-1	-1	-2	-2	-2	-3
C3	-1	-1	4	-2	-3	-3	-3	-4
C4	-1	-2	-2	8	-2	-3	-3	-3
C5	0	0	0	0	16	-1	-1	-1
C6	0	-1	-2	-2	-2	32	-3	-3
C7	0	0	-1	-2	-3	-3	64	-4
C8	0	-1	-2	-2	-2	-3	-3	128

Glass: This dataset consists of attributes of glass samples taken from the scan of an accident. The glass dataset contains 214 classes which belong to one of the six classes available in the domain. There is a total of 9 features and all of them are continuous. Benefit matrix favoring rare classes in this domain is given in Table A.3. Subsequently, the random matrix that has been used in experiments is presented in Table A.4.

Table A.3: Benefit table of glass dataset computed by using class probabilities in a way to favor minority class prediction

	Actual Class					
Prediction	C1	C2	C3	C4	C5	C6
C1	3	-1	-4	-5	-8	-2
C2	-1	2	-4	-6	-8	-3
C3	-1	-1	6	-2	-3	-1
C4	-1	-1	-1	8	-2	-1
C5	-0.5	-0.5	-1	-1	10	-1
C6	-2	-1	-2	-3	-4	5

Table A.4: Random benefit table of glass dataset with a ratio of 2 between consecutive class labels.

	Actual Class					
Prediction	C1	C2	C3	C4	C5	C6
C1	1	0	-1	-1	-2	-3
C2	0	2	0	-1	-2	-2
C3	-1	-2	4	-2	-3	-3
C4	-1	-2	-2	8	-2	-2
C5	0	-1	-1	-1	16	-2
C6	-1	-2	-2	-3	-3	32

Page-blocks: Compiled by Donato Malerba from Dipartimento di Informatica, University of Bari, The problem consists in classifying all the blocks of the page layout of a document that has been detected by a segmentation process. This is an essential step in document analysis in order to separate text from graphic areas. Indeed, the five classes are: text (1), horizontal line (2), picture (3), vertical line (4) and graphic (5). It is a relatively large database with 5473 examples from 54 distinct documents. All attributes are numeric with no missing values.

Table A.5: Benefit table of glass dataset dependent on class probabilities in inverse proportion such that minority class prediction is preferable.

	Actual Class				
Prediction	C1	C2	C3	C4	C5
C1	10	-15	-175	-56	-43
C2	-1	40	-12	-4	-3
C3	-1	-1	100	-1	-1
C4	-1	-2	-5	60	-1
C5	-1	-3	-10	-4	50

Table A.6: Random benefit table of glass dataset with a ratio of 3 between consecutive class labels.

	Actual Class				
Prediction	C1	C2	C3	C4	C5
C1	1	-2	-4	-5	-5
C2	-2	3	-3	-3	-4
C3	-1	-2	9	-5	-6
C4	-1	-3	-5	27	-6
C5	-1	-2	-3	-5	81

Vehicle: This dataset comes from the Turing Institute, Glasgow, Scotland. The purpose is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. There are 18 features with a total of 846 instances. Four types of vehicles to be classified are Opel, Saab, Bus and Van.

Table A.7: Benefit table of vehicle dataset dependent on class probabilities in inverse proportion such that minority class prediction is preferable.

	Actual Class			
Prediction	C1	C2	C3	C4
C1	4	-1	-2	-3
C2	-2	3	-1	-2
C3	-4	-2	2	-1
C4	-6	-3	-1	1

Table A.8: Random benefit table of vehicle dataset with a ratio of 3 between consecutive class labels.

	Actual Class			
Prediction	C1	C2	C3	C4
C1	1	-4	-4	-6
C2	-1	3	-2	-2
C3	-1	-3	9	-5
C4	-2	-3	-4	27

Wine: This dataset has been compiled by Institute of Pharmaceutical and Food Analysis and Technologies of Italy and it is on the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. In a classification context, this is a well posed problem with "well behaved" class structures. It is assumed to be a good data set for first testing of a new classifier and many other classification algorithms have used it for testing. All three classes are separable and there is a total of 178 instances and 13 attributes. Corresponding benefit tables we have used are shown in Table A.9 and Table A.10.

Table A.9: Benefit table of wine dataset dependent on class probabilities in inverse proportion such that minority class prediction is preferable.

	Actual Class		
Prediction	C1	C2	C3
C1	17	-8	-12
C2	-12	14	-15
C3	-8	-6	21

Table A.10: Randomly assigned benefits of wine dataset with a ratio of 4 between consecutive class labels.

	Actual Class		
Prediction	C1	C2	C3
C1	1	-3	-5
C2	-2	4	-8
C3	-1	-4	16

Appendix B

Special Datasets

There are five additional domains in which we have tested our BMFI algorithm. These datasets are called "special" because they have a specific benefit matrix assigned to them by domain experts or by hand, after a careful evaluation of the characteristics of the domains.

Arrhythmia2r: The data set used here consists of 526 ECG recordings [28]. Each record consists of a set of clinical parameters measured on rest ECG signals automatically by a commercially available system, and some personal information about the subjects. There are 279 parameters (features) in a single record. The patient population is divided into two groups based on the investigation of an expert cardiologist, as normal and abnormal. There are 246 cases in the normal group and 280 cases in the abnormal group. Out of 279 features 206 of them are continuous valued (linear) and 73 features are Boolean valued (nominal). 0.33% of the feature values are missing. Benefit table of the domain is adjusted as follows:

Table B.1: Benefit table of the arrhythmia2r dataset

	Actual Class	
Prediction	C1	C2
C1	1	-7
C2	-2	5

Bank-loans: The raw form of this dataset has been compiled by a private Turkish bank. We have preprocessed it by eliminating redundant data and missing attribute values. The entire raw dataset consists of more than 24000 instances. In this study, we have investigated a small, yet representative portion of it consisting 1443 instances. In the literature of machine learning, this data has been investigated initially in [33]. There are 13 attributes in the domain, 7 of them are linear and 6 are categorical. The dataset purpose is to predict whether a customer is likely to default or not. This domain is also very suitable for investigating the effects of feature dependency as presented in Section 5.5. The static benefit table used for testing is given in Table B.2.

Table B.2: Benefit table of the bank-loans dataset

	Actual Class	
Prediction	C1	C2
C1	4	-20
C2	-4	0

Bankruptcy: This is the data compiled from the Compact Disclosures™ by Dorsey et al.[32]. The data set consists of financial ratios from firms for three successive years, 1989--1991. Each case (instance) in the data set contains the values of the 19 ratios about a firm for a year and an indicator of whether or not the firm failed in the following year. Here the failure is defined as financial distress. A firm is in financial distress if it has entered bankruptcy under chapters 7 or 11 of the U.S. Bankruptcy code. The data set contains 1444 instances, 414 of which are bankrupt firms. There are no missing values and the corresponding benefit table is given in Table B.3.

Table B.3: Benefit table of the bankruptcy dataset

	Actual Class	
Prediction	C1	C2
C1	1	-15
C2	-1	30

Dermatology: The differential diagnosis of erythemato-squamous diseases is a difficult problem in dermatology. They all share the clinical features of erythema and scaling, with very little differences. The diseases in this classification problem are psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, chronic dermatitis and pityriasis rubra pilaris. Patients were first evaluated clinically with 12 features. Skin samples were taken

for the evaluation of 22 histopathological features. Hence, there is a total of 34 features in the domain and only one of them is a categorical feature. The entire dataset consists of 366 patient records. For more information, please refer to [31].

Table B.4: Benefit table of the dermatology dataset

	Actual Class					
Prediction	C1	C2	C3	C4	C5	C6
C1	10	-5	-4	-5	-5	-4
C2	-3	6	-3	-4	-1	-3
C3	-2	-5	10	-5	-5	-4
C4	-5	-4	-4	8	-3	-3
C5	-3	-1	-2	-3	3	-2
C6	-2	-2	-2	-3	-3	4

Lesion (Gastric Carcinoma): This is the dataset of stomach cancer instances. There are 285 cases and all of them are malignant. The aim is to differentiate between stages of cancer. There are 9 classes representing the relative degree of the cancer. The first 4 classes are defined as early cancers whereas the remaining are counted as late cancers. Hence, the correct prediction of early cancer instances is more beneficial. Benefit table displaying such traits is given in Table B.5. There are 68 features, and only 7 of them are linear. In the entire dataset there are 970 missing features, which means that 5% of the dataset is missing. More information about the dataset is presented in [21].

Table B.5: Benefit table of the lesion dataset

	Actual Class								
Prediction	C1	C2	C3	C4	C5	C6	C7	C8	C9
C1	18	8	8	8	2	-10	-12	-15	-18
C2	10	15	12	12	4	-8	-10	-13	-15
C3	10	12	15	12	4	-8	-10	-13	-15
C4	10	12	12	15	4	-8	-10	-13	-15
C5	6	8	8	8	10	-6	-8	-11	-13
C6	-2	-3	-3	-3	-1	8	1	1	-2
C7	-4	-5	-5	-5	-3	4	6	3	-1
C8	-8	-10	-10	-10	-7	2	3	4	1
C9	-10	-12	-12	-12	-9	1	1	2	3